

MLBookProc

1.2.1

Generated by Doxygen 1.15.0

1 MLBookProc	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Class Documentation	7
4.1 AddBook Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 AddBook()	8
4.1.3 Member Function Documentation	8
4.1.3.1 add_to_existing_archive()	8
4.1.3.2 add_to_existing_archive_dir()	8
4.1.3.3 archive_filenames()	9
4.1.3.4 overwrite_archive()	9
4.1.3.5 overwrite_archive_dir()	9
4.1.3.6 simple_add()	10
4.1.3.7 simple_add_dir()	10
4.2 ArchEntry Class Reference	11
4.2.1 Detailed Description	11
4.3 ArchiveFileEntry Class Reference	11
4.3.1 Detailed Description	12
4.4 ArchiveRemoveEntry Class Reference	12
4.4.1 Detailed Description	13
4.4.2 Member Function Documentation	13
4.4.2.1 reset()	13
4.5 ARCHParser Class Reference	13
4.5.1 Detailed Description	15
4.5.2 Constructor & Destructor Documentation	15
4.5.2.1 ARCHParser()	15
4.5.3 Member Function Documentation	16
4.5.3.1 arch_parser()	16
4.6 AuxFunc Class Reference	16
4.6.1 Detailed Description	18
4.6.2 Member Function Documentation	18
4.6.2.1 copy_book_callback()	18
4.6.2.2 create()	18
4.6.2.3 detect_encoding()	18
4.6.2.4 get_activated()	19
4.6.2.5 get_charset_conv_quantity()	19

4.6.2.6	get_converter_by_number()	19
4.6.2.7	get_extension()	19
4.6.2.8	get_genre_list()	20
4.6.2.9	get_selfpath()	20
4.6.2.10	get_supported_archive_types_packing()	20
4.6.2.11	get_supported_archive_types_unpacking()	20
4.6.2.12	get_supported_types()	21
4.6.2.13	getDJVUContext()	21
4.6.2.14	homePath()	21
4.6.2.15	html_to_utf8()	21
4.6.2.16	if_supported_type()	22
4.6.2.17	ifSupportedArchivePackingType()	22
4.6.2.18	ifSupportedArchiveUnpackaingType()	22
4.6.2.19	libgcrypt_error_handling()	23
4.6.2.20	open_book_callback()	23
4.6.2.21	randomFileName()	24
4.6.2.22	share_path()	24
4.6.2.23	stringToLower()	24
4.6.2.24	temp_path()	24
4.6.2.25	time_t_to_date()	24
4.6.2.26	to_hex()	25
4.6.2.27	to_utf_8()	25
4.6.2.28	utf8_to_system()	25
4.6.2.29	utf_8_to()	26
4.7	BaseKeeper Class Reference	26
4.7.1	Detailed Description	27
4.7.2	Constructor & Destructor Documentation	27
4.7.2.1	BaseKeeper()	27
4.7.3	Member Function Documentation	27
4.7.3.1	booksWithNotes()	27
4.7.3.2	collectionAuthors()	28
4.7.3.3	get_base_vector()	28
4.7.3.4	get_books_path()	28
4.7.3.5	getBooksQuantity()	29
4.7.3.6	loadCollection()	29
4.7.3.7	searchBook()	29
4.7.4	Member Data Documentation	30
4.7.4.1	auth_show_progr	30
4.8	BookBaseEntry Class Reference	30
4.8.1	Detailed Description	31
4.8.2	Constructor & Destructor Documentation	31
4.8.2.1	BookBaseEntry()	31

4.9 BookInfo Class Reference	32
4.9.1 Detailed Description	32
4.9.2 Constructor & Destructor Documentation	32
4.9.2.1 BookInfo()	32
4.9.3 Member Function Documentation	32
4.9.3.1 get_book_info()	32
4.9.3.2 set_dpi()	33
4.10 BookInfoEntry Class Reference	33
4.10.1 Detailed Description	34
4.10.2 Member Enumeration Documentation	34
4.10.2.1 cover_types	34
4.11 BookMarks Class Reference	35
4.11.1 Detailed Description	35
4.11.2 Constructor & Destructor Documentation	35
4.11.2.1 BookMarks()	35
4.11.3 Member Function Documentation	36
4.11.3.1 createBookMark()	36
4.11.3.2 getBookMarks()	36
4.11.3.3 removeBookMark()	36
4.12 BookParseEntry Class Reference	37
4.12.1 Detailed Description	37
4.12.2 Member Data Documentation	38
4.12.2.1 book_genre	38
4.12.2.2 book_name	38
4.12.2.3 book_path	38
4.13 ByteOrder Class Reference	38
4.13.1 Detailed Description	40
4.13.2 Constructor & Destructor Documentation	40
4.13.2.1 ByteOrder() [1/8]	40
4.13.2.2 ByteOrder() [2/8]	40
4.13.2.3 ByteOrder() [3/8]	41
4.13.2.4 ByteOrder() [4/8]	41
4.13.2.5 ByteOrder() [5/8]	41
4.13.2.6 ByteOrder() [6/8]	41
4.13.2.7 ByteOrder() [7/8]	42
4.13.2.8 ByteOrder() [8/8]	42
4.13.3 Member Function Documentation	42
4.13.3.1 get_big()	42
4.13.3.2 get_little()	42
4.13.3.3 get_native()	43
4.13.3.4 operator=() [1/8]	43
4.13.3.5 operator=() [2/8]	43

4.13.3.6 operator=() [3/8]	43
4.13.3.7 operator=() [4/8]	44
4.13.3.8 operator=() [5/8]	44
4.13.3.9 operator=() [6/8]	44
4.13.3.10 operator=() [7/8]	44
4.13.3.11 operator=() [8/8]	45
4.13.3.12 set_big()	45
4.13.3.13 set_little()	45
4.14 CreateCollection Class Reference	46
4.14.1 Detailed Description	48
4.14.2 Constructor & Destructor Documentation	48
4.14.2.1 CreateCollection() [1/2]	48
4.14.2.2 CreateCollection() [2/2]	49
4.14.3 Member Function Documentation	49
4.14.3.1 closeBaseFile()	49
4.14.3.2 createCollection()	49
4.14.3.3 openBaseFile()	50
4.14.3.4 threadRegulator()	50
4.14.3.5 write_file_to_base()	50
4.14.4 Member Data Documentation	50
4.14.4.1 already_hashed	50
4.14.4.2 base_path	51
4.14.4.3 books_path	51
4.14.4.4 current_bytes	51
4.14.4.5 need_to_parse	51
4.14.4.6 progress	51
4.14.4.7 pulse	52
4.14.4.8 rar_support	52
4.14.4.9 signal_total_bytes	52
4.15 DCParse Class Reference	52
4.15.1 Detailed Description	54
4.15.2 Constructor & Destructor Documentation	54
4.15.2.1 DCParse()	54
4.15.3 Member Function Documentation	54
4.15.3.1 dcAuthor()	54
4.15.3.2 dcDate()	55
4.15.3.3 dcDescription()	55
4.15.3.4 dcGenre()	55
4.15.3.5 dcIdentifier()	56
4.15.3.6 dcLanguage()	56
4.15.3.7 dcPublisher()	57
4.15.3.8 dcSource()	57

4.15.3.9 dcTitle()	57
4.16 DJVUParser Class Reference	58
4.16.1 Detailed Description	58
4.16.2 Constructor & Destructor Documentation	58
4.16.2.1 DJVUParser()	58
4.16.3 Member Function Documentation	58
4.16.3.1 djvu_book_info()	58
4.16.3.2 djvu_parser()	59
4.17 ElectroBookInfoEntry Class Reference	59
4.17.1 Detailed Description	60
4.18 EPUBParser Class Reference	60
4.18.1 Detailed Description	63
4.18.2 Constructor & Destructor Documentation	63
4.18.2.1 EPUBParser()	63
4.18.3 Member Function Documentation	63
4.18.3.1 epub_book_info()	63
4.18.3.2 epub_parser()	63
4.19 FB2Parser Class Reference	64
4.19.1 Detailed Description	65
4.19.2 Constructor & Destructor Documentation	65
4.19.2.1 FB2Parser()	65
4.19.3 Member Function Documentation	66
4.19.3.1 fb2_book_info()	66
4.19.3.2 fb2_parser()	66
4.20 FileParseEntry Class Reference	67
4.20.1 Detailed Description	67
4.20.2 Member Data Documentation	67
4.20.2.1 books	67
4.20.2.2 file_hash	68
4.20.2.3 file_rel_path	68
4.21 FormatAnnotation Class Reference	68
4.21.1 Detailed Description	69
4.21.2 Constructor & Destructor Documentation	69
4.21.2.1 FormatAnnotation()	69
4.21.3 Member Function Documentation	69
4.21.3.1 final_cleaning()	69
4.21.3.2 remove_escape_sequences()	70
4.21.3.3 removeAllTags()	70
4.21.3.4 replace_tags()	70
4.21.3.5 setTagReplacementTable()	71
4.22 Genre Class Reference	71
4.22.1 Detailed Description	72

4.22.2 Member Data Documentation	72
4.22.2.1 genre_code	72
4.22.2.2 genre_name	72
4.23 GenreGroup Class Reference	72
4.23.1 Detailed Description	73
4.23.2 Member Data Documentation	73
4.23.2.1 group_name	73
4.24 Hasher Class Reference	73
4.24.1 Detailed Description	74
4.24.2 Constructor & Destructor Documentation	74
4.24.2.1 Hasher()	74
4.24.3 Member Function Documentation	74
4.24.3.1 buf_hashing()	74
4.24.3.2 file_hashing()	75
4.24.4 Member Data Documentation	75
4.24.4.1 cancel	75
4.24.4.2 stop_all_signal	76
4.25 LibArchive Class Reference	76
4.25.1 Detailed Description	77
4.25.2 Constructor & Destructor Documentation	78
4.25.2.1 LibArchive()	78
4.25.3 Member Function Documentation	78
4.25.3.1 createArchFile()	78
4.25.3.2 fileinfo()	78
4.25.3.3 fileNames()	79
4.25.3.4 fileNamesStream()	79
4.25.3.5 libarchive_error()	80
4.25.3.6 libarchive_packing() [1/2]	80
4.25.3.7 libarchive_packing() [2/2]	81
4.25.3.8 libarchive_read_entry()	81
4.25.3.9 libarchive_read_entry_str()	81
4.25.3.10 libarchive_read_init()	82
4.25.3.11 libarchive_read_init_fallback()	82
4.25.3.12 libarchive_remove_entry()	82
4.25.3.13 libarchive_remove_init()	83
4.25.3.14 libarchive_write_data()	83
4.25.3.15 libarchive_write_data_from_file()	84
4.25.3.16 libarchive_write_directory()	84
4.25.3.17 libarchive_write_file()	85
4.25.3.18 libarchive_write_init()	85
4.25.3.19 unpackByFileNameStream()	85
4.25.3.20 unpackByFileNameStreamStr()	86

4.25.3.21 unpackByPosition()	86
4.25.3.22 unpackByPositionStr()	87
4.25.3.23 writeToArchive()	87
4.26 MException Class Reference	88
4.26.1 Detailed Description	88
4.26.2 Constructor & Destructor Documentation	88
4.26.2.1 MException()	88
4.26.3 Member Function Documentation	89
4.26.3.1 what()	89
4.27 NotesBaseEntry Class Reference	89
4.27.1 Detailed Description	90
4.27.2 Constructor & Destructor Documentation	90
4.27.2.1 NotesBaseEntry()	90
4.28 NotesKeeper Class Reference	90
4.28.1 Detailed Description	91
4.28.2 Constructor & Destructor Documentation	91
4.28.2.1 NotesKeeper()	91
4.28.3 Member Function Documentation	91
4.28.3.1 editNote()	91
4.28.3.2 getNote()	92
4.28.3.3 getNotesForCollection()	92
4.28.3.4 loadBase()	93
4.28.3.5 readNote()	93
4.28.3.6 readNoteText()	93
4.28.3.7 refreshCollection()	93
4.28.3.8 removeCollection()	94
4.28.3.9 removeNotes()	94
4.29 ODTParser Class Reference	95
4.29.1 Detailed Description	97
4.29.2 Constructor & Destructor Documentation	97
4.29.2.1 ODTParser()	97
4.29.3 Member Function Documentation	98
4.29.3.1 odtBookInfo()	98
4.29.3.2 odtParser()	98
4.30 OmpLockGuard Class Reference	99
4.30.1 Detailed Description	99
4.30.2 Constructor & Destructor Documentation	99
4.30.2.1 OmpLockGuard()	99
4.31 OpenBook Class Reference	100
4.31.1 Detailed Description	100
4.31.2 Constructor & Destructor Documentation	100
4.31.2.1 OpenBook()	100

4.31.3 Member Function Documentation	100
4.31.3.1 open_book()	100
4.32 PaperBookInfoEntry Class Reference	101
4.32.1 Detailed Description	102
4.33 PDFParser Class Reference	102
4.33.1 Detailed Description	102
4.33.2 Constructor & Destructor Documentation	102
4.33.2.1 PDFParser()	102
4.33.3 Member Function Documentation	102
4.33.3.1 pdf_annotation_n_cover()	102
4.33.3.2 pdf_parser()	103
4.34 RefreshCollection Class Reference	103
4.34.1 Detailed Description	106
4.34.2 Constructor & Destructor Documentation	106
4.34.2.1 RefreshCollection()	106
4.34.3 Member Function Documentation	107
4.34.3.1 editBook()	107
4.34.3.2 refreshBook()	107
4.34.3.3 refreshCollection()	108
4.34.3.4 refreshFile()	108
4.34.3.5 set_rar_support()	108
4.34.4 Member Data Documentation	109
4.34.4.1 bytes_hashed	109
4.34.4.2 total_bytes_to_hash	109
4.35 RemoveBook Class Reference	109
4.35.1 Detailed Description	109
4.35.2 Constructor & Destructor Documentation	109
4.35.2.1 RemoveBook()	109
4.35.3 Member Function Documentation	110
4.35.3.1 removeBook()	110
4.36 ReplaceTagItem Class Reference	110
4.36.1 Detailed Description	111
4.37 SelfRemovingPath Class Reference	111
4.37.1 Detailed Description	111
4.37.2 Constructor & Destructor Documentation	111
4.37.2.1 SelfRemovingPath()	111
4.37.3 Member Function Documentation	112
4.37.3.1 operator=()	112
4.38 TXTParser Class Reference	112
4.38.1 Detailed Description	112
4.38.2 Constructor & Destructor Documentation	112
4.38.2.1 TXTParser()	112

4.38.3 Member Function Documentation	113
4.38.3.1 txtBookInfo()	113
4.38.3.2 txtParser()	113
4.39 XMLParser Class Reference	114
4.39.1 Detailed Description	114
4.39.2 Constructor & Destructor Documentation	114
4.39.2.1 XMLParser()	114
4.39.3 Member Function Documentation	115
4.39.3.1 get_book_encoding()	115
4.39.3.2 get_element_attribute()	115
4.39.3.3 get_tag()	115
4.39.3.4 htmlSymbolsReplacement()	116
4.39.3.5 listAllTags()	116
4.39.3.6 removeAllTags()	116
4.39.3.7 searchTag()	117
4.40 XMLTag Class Reference	117
4.40.1 Detailed Description	118
4.40.2 Member Function Documentation	118
4.40.2.1 hasContent()	118
4.40.3 Member Data Documentation	118
4.40.3.1 content_end	118
4.40.3.2 content_start	118
4.40.3.3 element	118
Index	119

Chapter 1

MLBookProc

MLBookProc is a library for managing `.fb2`, `.epub`, `.pdf`, `.djvu` e-books, `.odt`, `.txt` and `.md` files collections. It can also works with same formats packed in `zip`, `7z`, `jar`, `cpio`, `iso`, `tar`, `tar.gz`, `tar.bz2`, `tar.xz`, `rar` archives (`rar` archives are available only for reading) itself or packed in same types of archives with `.fbd` files (any files, not only books). **MLBookProc** creates own database and does not change files content, names or location.

To start add cmake package MLBookProc to your project.

```
find_package(MLBookProc)
if(MLBookProc_FOUND)
    target_link_libraries(myproject MLBookProc::mlbookproc)
endif()
```

Note

MLBookProc sets `USE_OPENMP` build variable in case of OpenMP usage.

Then create [AuxFunc](#) object. Further reading: [CreateCollection](#), [RefreshCollection](#), [BaseKeeper](#), [BookMarks](#), [NotesKeeper](#), [BookInfo](#), [OpenBook](#).

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AddBook	7
ArchEntry	11
ArchiveFileEntry	11
ArchiveRemoveEntry	12
AuxFunc	16
BaseKeeper	26
BookBaseEntry	30
BookInfo	32
BookInfoEntry	33
BookMarks	35
BookParseEntry	37
ByteOrder	38
DJVUParser	58
ElectroBookInfoEntry	59
FileParseEntry	67
Genre	71
GenreGroup	72
Hasher	73
CreateCollection	46
RefreshCollection	103
LibArchive	76
ARCHParser	13
EPUBParser	60
ODTParser	95
MLException	88
NotesBaseEntry	89
NotesKeeper	90
OmpLockGuard	99
OpenBook	100
PaperBookInfoEntry	101
PDFParser	102
RemoveBook	109
ReplaceTagItem	110
SelfRemovingPath	111
TXTParser	112

XMLParser	114
DCParser	52
EPUBParser	60
FB2Parser	64
FormatAnnotation	68
ODTParser	95
XMLTag	117

Chapter 3

Class Index

3.1 Class List

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

AddBook	
The AddBook class	7
ArchEntry	
The ArchEntry class	11
ArchiveFileEntry	
The ArchiveFileEntry class	11
ArchiveRemoveEntry	
The ArchiveRemoveEntry class	12
ARCHParser	
The ARCHParser class	13
AuxFunc	
The AuxFunc class	16
BaseKeeper	
The BaseKeeper class	26
BookBaseEntry	
The BookBaseEntry class	30
BookInfo	
The BookInfo class	32
BookInfoEntry	
The BookInfoEntry class	33
BookMarks	
The BookMarks class	35
BookParseEntry	
The BookParseEntry class	37
ByteOrder	
The ByteOrder class	38
CreateCollection	
The CreateCollection class	46
DCParser	
The DCParser class	52
DJVUParser	
The DJVUParser class	58
ElectroBookInfoEntry	
The ElectroBookInfoEntry class	59
EPUBParser	
The EPUBParser class	60

FB2Parser	
The FB2Parser class	64
FileParseEntry	
The FileParseEntry class	67
FormatAnnotation	
The FormatAnnotation class	68
Genre	
The Genre class	71
GenreGroup	
The GenreGroup class	72
Hasher	
The Hasher class	73
LibArchive	
The LibArchive class	76
MLException	
The MLException class	88
NotesBaseEntry	
The NotesBaseEntry class	89
NotesKeeper	
The NotesKeeper class	90
ODTParser	
The ODTParser class	95
OmpLockGuard	
The OmpLockGuard class	99
OpenBook	
The OpenBook class	100
PaperBookInfoEntry	
The PaperBookInfoEntry class	101
PDFParser	
The PDFParser class	102
RefreshCollection	
The RefreshCollection class	103
RemoveBook	
The RemoveBook class	109
ReplaceTagItem	
The ReplaceTagItem class	110
SelfRemovingPath	
The SelfRemovingPath class	111
TXTParser	
The TXTParser class	112
XMLParser	
The XMLParser class	114
XMLTag	
The XMLTag class	117

Chapter 4

Class Documentation

4.1 AddBook Class Reference

The [AddBook](#) class.

```
#include <AddBook.h>
```

Public Member Functions

- [AddBook](#) (const std::shared_ptr< [AuxFunc](#) > &af, const std::string &collection_name, const bool &remove_←
_sources, const std::shared_ptr< [BookMarks](#) > &bookmarks)
AddBook constructor.
- void [simple_add](#) (const std::vector< std::tuple< std::filesystem::path, std::filesystem::path > > &books)
Adds single book files.
- void [simple_add_dir](#) (const std::vector< std::tuple< std::filesystem::path, std::filesystem::path > > &books)
Adds directories containing book files to collection.
- void [overwrite_archive](#) (const std::filesystem::path &archive_path, const std::vector< std::tuple< std←
::filesystem::path, std::filesystem::path > > &books)
Removes archive from collection and replaces it by new one.
- void [overwrite_archive_dir](#) (const std::filesystem::path &archive_path, const std::vector< std::tuple< std←
::filesystem::path, std::filesystem::path > > &books)
Removes archive from collection and replaces it by new one.
- void [add_to_existing_archive](#) (const std::filesystem::path &archive_path, const std::vector< std::tuple< std←
::filesystem::path, std::filesystem::path > > &books)
Adds books to existing archive.
- void [add_to_existing_archive_dir](#) (const std::filesystem::path &archive_path, const std::vector< std::tuple<←
std::filesystem::path, std::filesystem::path > > &books)
Adds books to existing archive.

Static Public Member Functions

- static std::vector< std::string > [archive_filenames](#) (const std::filesystem::path &archive_path, const std←
::shared_ptr< [AuxFunc](#) > &af)
Lists all files in archive.

4.1.1 Detailed Description

The [AddBook](#) class.

[AddBook](#) contains various methods to add books to collections.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AddBook()

```
AddBook::AddBook (
    const std::shared_ptr< AuxFunc > & af,
    const std::string & collection_name,
    const bool & remove_sources,
    const std::shared_ptr< BookMarks > & bookmarks)
```

[AddBook](#) constructor.

Parameters

<i>af</i>	AuxFunc object.
<i>collection_name</i>	name of collection book to be added to.
<i>remove_sources</i>	if <i>true</i> , MLBookProc will remove source file after book has been added.
<i>bookmarks</i>	BookMarks object.

4.1.3 Member Function Documentation

4.1.3.1 add_to_existing_archive()

```
void AddBook::add_to_existing_archive (
    const std::filesystem::path & archive_path,
    const std::vector< std::tuple< std::filesystem::path, std::filesystem::path > >
    & books)
```

Adds books to existing archive.

Unpacks existing archive and packs it again with old and new books inside.

Parameters

<i>archive_path</i>	absolute path to archive in collection.
<i>books</i>	vector, containing absolute paths to source books files (first tuple element) and relative paths to books in archive (second tuple element).

4.1.3.2 add_to_existing_archive_dir()

```
void AddBook::add_to_existing_archive_dir (
```

Parameters

<i>archive_path</i>	absolute path to archive in collection.
<i>books</i>	vector, containing absolute paths to source directories (first tuple element) and relative paths to directories in archive (second tuple element).

4.1.3.3 archive_filenames()

```
std::vector< std::string > AddBook::archive_filenames (
    const std::filesystem::path & archive_path,
    const std::shared_ptr< AuxFunc > & af) [static]
```

Lists all files in archive.

Parameters

<i>archive_path</i>	absolute path to archive.
<i>af</i>	smart pointer to AuxFunc object.

Returns

Vector containing relative paths of archive files.

4.1.3.4 overwrite_archive()

```
void AddBook::overwrite_archive (
    const std::filesystem::path & archive_path,
    const std::vector< std::tuple< std::filesystem::path, std::filesystem::path > >
    & books)
```

Removes archive from collection and replaces it by new one.

Removes archive from collection (if it exists) and creates new archive with the same name, containing books from books vector.

Parameters

<i>archive_path</i>	absolute path to archive file in collection.
<i>books</i>	vector, containing absolute paths to source books files (first tuple element) and relative paths to books files in archive (second tuple element).

4.1.3.5 overwrite_archive_dir()

Generated by Doxygen

```
void AddBook::overwrite_archive_dir (
    const std::filesystem::path & archive_path,
    const std::vector< std::tuple< std::filesystem::path, std::filesystem::path > >
```

Parameters

<i>archive_path</i>	absolute path to archive file in collection.
<i>books</i>	vector, containing absolute paths to source directories (first tuple element) and relative paths to directories in archive (second tuple element).

4.1.3.6 simple_add()

```
void AddBook::simple_add (
    const std::vector< std::tuple< std::filesystem::path, std::filesystem::path > >
    & books)
```

Adds single book files.

Adds books, listed in vector, to collection and parse them. User must set both paths manually.

Warning

Second tuples paths must not be outside collection books directory.

Parameters

<i>books</i>	vector, containing absolute paths to source files (first tuple element) and absolute paths of books files in collection (second tuple element).
--------------	---

4.1.3.7 simple_add_dir()

```
void AddBook::simple_add_dir (
    const std::vector< std::tuple< std::filesystem::path, std::filesystem::path > >
    & books)
```

Adds directories containing book files to collection.

Same as [simple_add\(\)](#), but adds directories containing books to collection.

Warning

Second tuples paths must not be outside collection books directory.

Parameters

<i>books</i>	vector, containing absolute paths to source directories (first tuple element) and absolute paths of directories in collection (second tuple element).
--------------	---

4.2 ArchEntry Class Reference

The [ArchEntry](#) class.

```
#include <ArchEntry.h>
```

Public Member Functions

- **ArchEntry** ()
ArchEntry constructor.
- **ArchEntry** (const [ArchEntry](#) &other)
ArchEntry copy constructor.
- **ArchEntry** ([ArchEntry](#) &&other)
ArchEntry move constructor.
- [ArchEntry](#) & **operator=** (const [ArchEntry](#) &other)
operator =
- [ArchEntry](#) & **operator=** ([ArchEntry](#) &&other)
operator =

Public Attributes

- uint64_t **size** = 0
Size of unpacked entry object (if available, 0 otherwise).
- uint64_t **compressed_size** = 0
Size of compressed object (if available, 0 otherwise).
- int64_t **position** = -1
Position of entry in archive file (if available, -1 otherwise).
- std::string **filename**
Path to file or directory in archive.

4.2.1 Detailed Description

The [ArchEntry](#) class.

Auxiliary class for [LibArchive](#). Contains some technical info about compressed files and directories. In most cases you do not need to create ArchEntry objects yourself.

4.3 ArchiveFileEntry Class Reference

The [ArchiveFileEntry](#) class.

```
#include <ArchiveFileEntry.h>
```

Public Member Functions

- **ArchiveFileEntry** ()
ArchiveFileEntry constructor.
- virtual ~**ArchiveFileEntry** ()
ArchiveFileEntry destructor.

Public Attributes

- `std::fstream file`
Archive file stream.
- `std::filesystem::path file_path`
Archive file absolute path.
- `la_ssize_t buf_sz = 1048576`
Size of buffer to be used in file stream.
- `la_ssize_t read_bytes = 0`
Number of bites already read from archive file.
- `la_ssize_t file_size = 0`
Archive file size.
- `char * read_buf = nullptr`
Buffer to be used in file stream.

4.3.1 Detailed Description

The [ArchiveFileEntry](#) class.

Auxiliary class for [LibArchive](#) methods. In most cases you do not need to create it directly, use [LibArchive::createArchFile\(\)](#) method instead.

4.4 ArchiveRemoveEntry Class Reference

The [ArchiveRemoveEntry](#) class.

```
#include <ArchiveRemoveEntry.h>
```

Public Member Functions

- **ArchiveRemoveEntry ()**
[ArchiveRemoveEntry](#) constructor.
- virtual **~ArchiveRemoveEntry ()**
[ArchiveRemoveEntry](#) destructor.
- **ArchiveRemoveEntry (const [ArchiveRemoveEntry](#) &other)**
[ArchiveRemoveEntry](#) copy constructor.
- **ArchiveRemoveEntry ([ArchiveRemoveEntry](#) &&other)**
[ArchiveRemoveEntry](#) move constructor.
- **[ArchiveRemoveEntry](#) & operator= (const [ArchiveRemoveEntry](#) &other)**
operator =
- **[ArchiveRemoveEntry](#) & operator= ([ArchiveRemoveEntry](#) &&other)**
operator =
- void **reset ()**
Resets all internal objects.

Public Attributes

- `std::shared_ptr< archive > a_read`
libarchive object, file to be removed from.
- `std::shared_ptr< archive > a_write`
libarchive object for new archive.
- `std::shared_ptr< ArchiveFileEntry > fl`
ArchiveFileEntry object, file to be removed from.

4.4.1 Detailed Description

The [ArchiveRemoveEntry](#) class.

Auxiliary class for [LibArchive](#), to be used in case of removing files from archives. In most cases you do not need to use it directly.

4.4.2 Member Function Documentation

4.4.2.1 reset()

```
void ArchiveRemoveEntry::reset ()
```

Resets all internal objects.

Warning

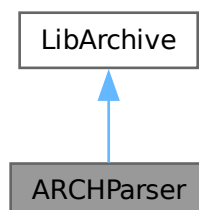
Do not use [ArchiveRemoveEntry](#) item after this method call!

4.5 ARCHParser Class Reference

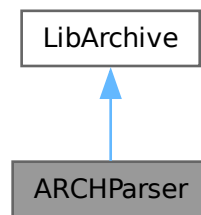
The [ARCHParser](#) class.

```
#include <ARCHParser.h>
```

Inheritance diagram for ARCHParser:



Collaboration diagram for ARCHParser:



Public Member Functions

- [ARCHParser](#) (const std::shared_ptr< [AuxFunc](#) > &af, const bool &rar_support)
ARCHParser constructor.
- virtual ~**ARCHParser** ()
ARCHParser destructor.
- std::vector< [BookParseEntry](#) > [arch_parser](#) (const std::filesystem::path &filepath)
This method carries out actual parsing.
- void **stopAll** ()
Stops all operations.

Public Member Functions inherited from [LibArchive](#)

- [LibArchive](#) (const std::shared_ptr< [AuxFunc](#) > &af)
LibArchive constructor.
- std::filesystem::path [unpackByPosition](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::string [unpackByPositionStr](#) (const std::filesystem::path &archaddress, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::filesystem::path [unpackByFileNameStream](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const std::string &filename)
Unpacks entry content from archive.
- std::string [unpackByFileNameStreamStr](#) (const std::filesystem::path &archaddress, const std::string &filename)
Unpacks entry content from archive.
- int [fileNames](#) (const std::filesystem::path &filepath, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.
- int [fileNamesStream](#) (const std::filesystem::path &address, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.
- [ArchEntry](#) [fileinfo](#) (const std::filesystem::path &address, const std::string &filename)
Returns ArchEntry for particular file or directory in archive.
- int [libarchive_packing](#) (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)
Packs file or directory into archive.

- `int libarchive_packing (const std::shared_ptr< archive > &a, const std::filesystem::path &sourcepath, const bool &rename_source, const std::string &new_source_name)`
Packs file or directory into archive.
- `ArchiveRemoveEntry libarchive_remove_init (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)`
Initializes archive objects for removing entries from archive.
- `int libarchive_remove_entry (ArchiveRemoveEntry rm_e, const std::vector< ArchEntry > &to_remove)`
Removes entry from archive.
- `void libarchive_error (const std::shared_ptr< archive > &a, const std::string &message, const int &error_number)`
Prints libarchive error messages.
- `std::string libarchive_read_entry_str (archive *a, archive_entry *entry)`
Reads archived file to string.
- `int libarchive_write_data (archive *a, const std::string &data)`
Writes data to archive.
- `std::shared_ptr< ArchiveFileEntry > createArchFile (const std::filesystem::path &archaddress, const la_int64_t &position=la_int64_t(0))`
Creates ArchiveFileEntry object.
- `std::shared_ptr< archive > libarchive_read_init (std::shared_ptr< ArchiveFileEntry > fl)`
Initializes archive reading.
- `std::shared_ptr< archive > libarchive_read_init_fallback (std::shared_ptr< ArchiveFileEntry > fl)`
Initializes archive reading.
- `std::filesystem::path libarchive_read_entry (archive *a, archive_entry *entry, const std::filesystem::path &outfolder)`
Unpacks libarchive entry content.
- `std::shared_ptr< archive > libarchive_write_init (const std::filesystem::path &outpath)`
Initializes writing to archive.
- `int writeToArchive (std::shared_ptr< archive > a, const std::filesystem::path &source, const std::filesystem::path &path_in_arch)`
Writes file or directory to archive.
- `int libarchive_write_directory (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch, const std::filesystem::path &source)`
Writes directory to archive.
- `int libarchive_write_file (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch, const std::filesystem::path &source)`
Writes file to archive.
- `int libarchive_write_data_from_file (archive *a, const std::filesystem::path &source)`
Writes raw data from file to archive.

4.5.1 Detailed Description

The [ARCHParser](#) class.

Class for archives parsing. In most cases you do not need to use this class directly. Use [CreateCollection](#) or [RefreshCollection](#) instead.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ARCHParser()

Generated by Doxygen

```
ARCHParser (
    const std::shared_ptr< AuxFunc > & af,
    const bool & rar_support)
```

Parameters

<i>af</i>	smart pointer to AuxFunc object.
<i>rar_support</i>	if <i>true</i> , ARCHParser will parse rar archives, otherwise not.

4.5.3 Member Function Documentation**4.5.3.1 arch_parser()**

```
std::vector< BookParseEntry > ARCHParser::arch_parser (
    const std::filesystem::path & filepath)
```

This method carries out actual parsing.

Call this method to start archive parsing.

Parameters

<i>filepath</i>	absolute path to archive.
-----------------	---------------------------

Returns

Vector of [BookParseEntry](#) objects.

4.6 AuxFunc Class Reference

The [AuxFunc](#) class.

```
#include <AuxFunc.h>
```

Public Member Functions

- virtual `~AuxFunc ()`
[AuxFunc](#) destructor.
- `std::string to_utf_8 (const std::string &input, const char *conv_name)`
Converts string to UTF-8 string.
- `std::string utf8_to_system (const std::string &input)`
Converts UTF-8 string to string in system default encoding.
- `std::string utf_8_to (const std::string &input, const char *conv_name)`
Converts UTF-8 string to string in chosen encoding.
- `const char * get_converter_by_number (const int32_t &num)`
Returns converter name.
- `std::string detect_encoding (const std::string &buf)`
Tries to detect string encoding.

- std::filesystem::path [homePath](#) ()
Returns user home directory path.
- std::filesystem::path [get_selfpath](#) ()
Returns absolute path to program executable file.
- std::filesystem::path [temp_path](#) ()
Returns absolute path to system temporary directory.
- std::filesystem::path [share_path](#) ()
*Returns absolute path to share directory, used by **MLBookProc**.*
- std::vector< [GenreGroup](#) > [get_genre_list](#) ()
Returns translated genre groups and genres names.
- std::string [libgcrypt_error_handling](#) (const gcry_error_t &err)
Auxiliary method to reinterpret libgcrypt errors as strings.
- std::string [to_hex](#) (const std::string &source)
Converts given string to hex format.
- std::string [stringToLower](#) (const std::string &line)
Converts all letters of the string to lowercase letters.
- std::string [randomFileName](#) ()
Returns random string.
- std::string [time_t_to_date](#) (const time_t &tt)
Converts time_t value to calendar date.
- bool [if_supported_type](#) (const std::filesystem::path &ch_p)
*Checks if given file is supported by **MLBookProc**.*
- bool [ifSupportedArchiveUnpackaingType](#) (const std::filesystem::path &ch_p)
*Checks if given archive is supported by **MLBookProc**.*
- bool [ifSupportedArchivePackingType](#) (const std::filesystem::path &ch_p)
*Checks if given archive is supported by **MLBookProc** for packing.*
- void [html_to_utf8](#) (std::string &input)
Converst 'html' symbols to UTF-8 characters.
- void [open_book_callback](#) (const std::filesystem::path &path)
Opens given file in default system application.
- void [copy_book_callback](#) (const std::filesystem::path &source, const std::filesystem::path &out)
Replaces out file by source file.
- std::vector< std::string > [get_supported_types](#) ()
Returns supported file types.
- std::vector< std::string > [get_supported_archive_types_packing](#) ()
Same as [get_supported_types\(\)](#), but returns only archives types, available for packing.
- std::vector< std::string > [get_supported_archive_types_unpacking](#) ()
Same as [get_supported_types\(\)](#), but returns only archives types, available for unpacking.
- std::string [get_extension](#) (const std::filesystem::path &p)
Returns file extesion.
- int32_t [get_charset_conv_quantity](#) ()
Returns number of available converters.
- bool [get_activated](#) ()
Checks if dependencies have been successfully activated.
- std::tuple< std::shared_ptr< [ddjvu_context_t](#) >, std::shared_ptr< int > > [getDJVUContext](#) ()
Returns tuple with smart pointer to djvu context object and smart pointer to djvu context signal pipe.

Static Public Member Functions

- static std::shared_ptr< [AuxFunc](#) > [create](#) ()
Creates [AuxFunc](#) object.

4.6.1 Detailed Description

The [AuxFunc](#) class.

[AuxFunc](#) class contains various useful auxiliary methods. [AuxFunc](#) object must be created (see [create\(\)](#)) before using of any **MLBookProc** methods or classes. [create\(\)](#) method should be called only once per program. Also it is strongly recommended to call [get_activated\(\)](#) method immediately after [AuxFunc](#) object creation.

4.6.2 Member Function Documentation

4.6.2.1 [copy_book_callback\(\)](#)

```
void AuxFunc::copy_book_callback (
    const std::filesystem::path & source,
    const std::filesystem::path & out)
```

Replaces out file by source file.

This method acts like `std::filesystem::copy`. It was introduced due to MinGW bug (MinGW ignores `std::filesystem::copy_options::overwrite_existing`).

Parameters

<i>source</i>	file to be copied.
<i>out</i>	file to be replaced.

4.6.2.2 [create\(\)](#)

```
std::shared_ptr< AuxFunc > AuxFunc::create () [static]
```

Creates [AuxFunc](#) object.

This method must be called before using of any **MLBookProc** classes or methods.

Warning

This method should be called only once per program.

Returns

Smart pointer to [AuxFunc](#) object.

4.6.2.3 [detect_encoding\(\)](#)

```
std::string AuxFunc::detect_encoding (
    const std::string & buf)
```

Tries to detect string encoding.

Parameters

<i>buf</i>	string which encoding is to be detected.
------------	--

Returns

String containing encoding name.

4.6.2.4 `get_activated()`

```
bool AuxFunc::get_activated ()
```

Checks if dependencies have been successfully activated.

Returns

true if all dependencies have been successfully activated, *false* otherwise.

4.6.2.5 `get_charset_conv_quantity()`

```
int32_t AuxFunc::get_charset_conv_quantity ()
```

Returns number of available converters.

See [icu documentation](#) for details.

Returns

Signed 32-bit integer number of available convertrs.

4.6.2.6 `get_converter_by_number()`

```
const char * AuxFunc::get_converter_by_number (  
    const int32_t & num)
```

Returns converter name.

Parameters

<i>num</i>	converter number (see icu documentation for details).
------------	---

Returns

Pointer to string, containing converter name.
Generated by Doxygen

4.6.2.7 `get_extension()`

Parameters

<i>p</i>	absolute path to file.
----------	------------------------

Returns

File extension with beginning dot (looks like ".tar.gz").

4.6.2.8 `get_genre_list()`

```
std::vector< GenreGroup > AuxFunc::get_genre_list ()
```

Returns translated genre groups and genres names.

Resulting genre groups and genres will be in default system language, if translations are available, or in English.

Returns

Vector of [GenreGroup](#) objects.

4.6.2.9 `get_selfpath()`

```
std::filesystem::path AuxFunc::get_selfpath ()
```

Returns absolute path to program executable file.

Returns

Absolute path to executable file.

4.6.2.10 `get_supported_archive_types_packing()`

```
std::vector< std::string > AuxFunc::get_supported_archive_types_packing ()
```

Same as [get_supported_types\(\)](#), but returns only archives types, available for packing.

Returns

Vector of supported archives types.

4.6.2.11 `get_supported_archive_types_unpacking()`

```
std::vector< std::string > AuxFunc::get_supported_archive_types_unpacking ()
```

Same as [get_supported_types\(\)](#), but returns only archives types, available for unpacking.

Returns

Vector of supported archives types.

4.6.2.12 get_supported_types()

```
std::vector< std::string > AuxFunc::get_supported_types ()
```

Returns supported file types.

Returns

Vector of supported files extensions without beginning dot (looks like "fb2").

4.6.2.13 getDJVUContext()

```
std::tuple< std::shared_ptr< ddjvu_context_t >, std::shared_ptr< int > > AuxFunc::getDJVUContext ()
```

Returns tuple with smart pointer to djvu context object and smart pointer to djvu context signal pipe.

First element of returned tuple is the smart pointer to djvu context object. Second element is smart pointer to array, containing signal pipe file descriptors. This descriptors can be used to obtain signals, emitted by djvu context on new message existence (see [DjVuLibre](#) documentation). First element of array is a reading end of pipe (can be accessed by *read* OS function), second element is the writing end of pipe (can be accessed by *write* OS function).

Warning

On Windows you must convert pointer to pipe array to HANDLE pointer before use:

```
std::tuple<std::shared_ptr<ddjvu_context_t>, std::shared_ptr<int>>
context_tuple;

context_tuple = getDJVUContext();

HANDLE *handles = reinterpret_cast<HANDLE
*>(std::get<1>(context_tuple).get());
```

Warning

In case of multiply documents are processed by djvu context, pipe signals are emitted for all pipes at once. It means, that you must check document object on receiving signal by the pipe.

Returns

Tuple with the smart pointer to djvu context object and smart pointer to pipe file descriptors.

4.6.2.14 homePath()

```
std::filesystem::path AuxFunc::homePath ()
```

Returns user home directory path.

Returns

Absolute path to home directory.

4.6.2.15 html_to_utf8()

Parameters

<i>input</i>	string to be converted.
--------------	-------------------------

4.6.2.16 if_supported_type()

```
bool AuxFunc::if_supported_type (
    const std::filesystem::path & ch_p)
```

Checks if given file is supported by **MLBookProc**.

'Supported types' check is carried out by file extension.

Parameters

<i>ch↔ _p</i>	absolute path to file.
-------------------	------------------------

Returns

true if file is supported, *false* otherwise.

4.6.2.17 ifSupportedArchivePackingType()

```
bool AuxFunc::ifSupportedArchivePackingType (
    const std::filesystem::path & ch_p)
```

Checks if given archive is supported by **MLBookProc** for packing.

Same as [ifSupportedArchiveUnpackaingType\(\)](#), but checks if given file is supported for packing

Parameters

<i>ch↔ _p</i>	absolute path to archive (may not exist).
-------------------	---

Returns

true if archive is supported, *false* otherwise.

4.6.2.18 ifSupportedArchiveUnpackaingType()

```
bool AuxFunc::ifSupportedArchiveUnpackaingType (
    const std::filesystem::path & ch_p)
```

Parameters

<i>ch← _p</i>	absolute path to archive (may not exist).
-------------------	---

Returns

true if archive is supported, *false* otherwise.

4.6.2.19 libgcrypt_error_handling()

```
std::string AuxFunc::libgcrypt_error_handling (  
    const gcry_error_t & err)
```

Auxiliary method to reinterpret libgcrypt errors as strings.

In most cases you do not need to call this method directly.

Parameters

<i>err</i>	libgcrypt error code (see libgcrypt documentation for details).
------------	---

Returns

Human-readable string explaining error code.

4.6.2.20 open_book_callback()

```
void AuxFunc::open_book_callback (  
    const std::filesystem::path & path)
```

Opens given file in default system application.

Parameters

<i>path</i>	absolute path to file to be opened.
-------------	-------------------------------------

4.6.2.21 randomFileName()

```
std::string AuxFunc::randomFileName ()
```

Returns random string.

Returns

Random string (it will look like "<random_hex_numbe>MLBookProc").

4.6.2.22 share_path()

```
std::filesystem::path AuxFunc::share_path ()
```

Returns absolute path to *share* directory, used by **MLBookProc**.

Result path is calculating as path relative to program executable file path ('absolute_path_to_executable_↵ file/./share').

Returns

Absolute path to *share* directory, used by **MLBookProc**.

4.6.2.23 stringToLower()

```
std::string AuxFunc::stringToLower (
    const std::string & line)
```

Converts all letters of the string to lowercase letters.

Parameters

<i>line</i>	UTF-8 encoded string to be converted to lowercase.
-------------	--

Returns

Lowercase string.

4.6.2.24 temp_path()

```
std::filesystem::path AuxFunc::temp_path ()
```

Returns absolute path to system temporary directory.

Returns

Absolute path to system temporary directory.

Parameters

<i>tt</i>	time_t value.
-----------	---------------

Returns

Date string (it will look like "<day_number>.<month_number>.<year>")

4.6.2.26 to_hex()

```
std::string AuxFunc::to_hex (  
    const std::string & source)
```

Converts given string to hex format.

Each char element will be converted to two hexadecimal digits.

Parameters

<i>source</i>	string to be converted.
---------------	-------------------------

Returns

String in hex format.

4.6.2.27 to_utf_8()

```
std::string AuxFunc::to_utf_8 (  
    const std::string & input,  
    const char * conv_name)
```

Converts string to UTF-8 string.

Parameters

<i>input</i>	string to be converted.
<i>conv_name</i>	input string encoding name (see icu documentation for details).

Returns

UTF-8 encoded string or empty string in case of any error.

4.6.2.28 utf8_to_system()

Parameters

<i>input</i>	UTF-8 string to be converted.
--------------	-------------------------------

Returns

String in system encoding or empty string in case of any error.

4.6.2.29 utf_8_to()

```
std::string AuxFunc::utf_8_to (
    const std::string & input,
    const char * conv_name)
```

Converts UTF-8 string to string in chosen encoding.

Parameters

<i>input</i>	string to be converted.
<i>conv_name</i>	output string encoding name (see icu documentation for details).

Returns

String in chosen encoding or empty string in case of any error.

4.7 BaseKeeper Class Reference

The [BaseKeeper](#) class.

```
#include <BaseKeeper.h>
```

Public Member Functions

- [BaseKeeper](#) (const std::shared_ptr< [AuxFunc](#) > &af)
BaseKeeper constructor.
- virtual ~**BaseKeeper** ()
BaseKeeper destructor.
- void [loadCollection](#) (const std::string &col_name)
Loads collection database to memory.
- std::vector< [BookBaseEntry](#) > [searchBook](#) (const [BookBaseEntry](#) &search, const double &coef_↵
coincidence=double(0.7))
Searches book in collection.
- std::vector< std::string > [collectionAuthors](#) ()
Lists all authors, found in collection.

- `std::vector< BookBaseEntry > booksWithNotes` (const `std::vector< NotesBaseEntry > ¬es`)
Lists all books of current collection, which have notes.
- `void stopSearch ()`
Stops all search operations.
- `void clearBase ()`
Unloads collection base from memory.
- `std::vector< FileParseEntry > get_base_vector ()`
Returns copy of inner database vector.
- `size_t getBooksQuantity ()`
Returns total quantity of books in loaded collection.

Static Public Member Functions

- static `std::filesystem::path get_books_path` (const `std::string &collection_name`, const `std::shared_ptr< AuxFunc > &af`)
Returns absolute path to directory containing collection books.

Public Attributes

- `std::function< void(const double &progr, const double &sz)> auth_show_progr`
[collectionAuthors\(\)](#) progress callback

4.7.1 Detailed Description

The [BaseKeeper](#) class.

This class is intended to keep and operate collections databases. [loadCollection\(\)](#) method should be called first.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 BaseKeeper()

```
BaseKeeper::BaseKeeper (
    const std::shared_ptr< AuxFunc > & af)
```

[BaseKeeper](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.7.3 Member Function Documentation

4.7.3.1 booksWithNotes()

Generated by Doxygen

```
std::vector< BookBaseEntry > BaseKeeper::booksWithNotes (
    const std::vector< NotesBaseEntry > & notes)
```

Parameters

<i>notes</i>	vector of notes (see NotesKeeper class documentation).
--------------	--

Returns

Vector of books with notes.

4.7.3.2 collectionAuthors()

```
std::vector< std::string > BaseKeeper::collectionAuthors ()
```

Lists all authors, found in collection.

Returns

Vector containing UTF-8 author's names strings.

4.7.3.3 get_base_vector()

```
std::vector< FileParseEntry > BaseKeeper::get_base_vector ()
```

Returns copy of inner database vector.

Returns

Database vector.

4.7.3.4 get_books_path()

```
std::filesystem::path BaseKeeper::get_books_path (
    const std::string & collection_name,
    const std::shared_ptr< AuxFunc > & af) [static]
```

Returns absolute path to directory containing collection books.

This method can be called without collection loading to memory.

Note

This method can throw [MLException](#) in case of errors.

Parameters

<i>collection_name</i>	collection name.
<i>af</i>	smart pointer to AuxFunc object.

Returns

Absolute path to books directory.

4.7.3.5 getBooksQuantity()

```
size_t BaseKeeper::getBooksQuantity ()
```

Returns total quantity of books in loaded collection.

Note

If this method is called during collection loading, it will block calling thread execution until collection is loaded.

Returns

Total books quantity in loaded collection.

4.7.3.6 loadCollection()

```
void BaseKeeper::loadCollection (
    const std::string & col_name)
```

Loads collection database to memory.

Note

This method can throw [MLException](#) in case of errors.

Parameters

<i>col_name</i>	collection name.
-----------------	------------------

4.7.3.7 searchBook()

```
std::vector< BookBaseEntry > BaseKeeper::searchBook (
    const BookBaseEntry & search,
    const double & coef_coincedence = double(0.7))
```

Searches book in collection.

[BookBaseEntry](#) object must be provided as search request. It is necessary to fill in any field in the inner [BookParseEntry](#) object to receive particular result. Otherwise complete collection book list will be returned.

Parameters

<i>search</i>	BookBaseEntry object. If author search is needed bpe.book_author field must be in the following form: "Surname\7First Name\7Second name".
<i>coef_coincedence</i>	Required coefficient of coincidence for search result. Allowed values are from 0.0 to 1.0. Default value is 0.7. If value is greater then 1.0, it means "exact match". In that case this method will return only results with the exact match for all search fields (empty search parameters are ignored).

Returns

Vector of [BookBaseEntry](#) objects, containing search results.

4.7.4 Member Data Documentation

4.7.4.1 auth_show_progr

```
std::function<void(const double &progr, const double &sz)> BaseKeeper::auth_show_progr
```

[collectionAuthors\(\)](#) progress callback

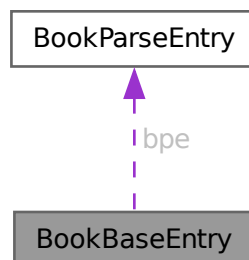
[collectionAuthors\(\)](#) method execution can take some time. This callback indicates progress. *progr* is current progress in conventional units. *sz* is total quantity of conventional units to be processed.

4.8 BookBaseEntry Class Reference

The [BookBaseEntry](#) class.

```
#include <BookBaseEntry.h>
```

Collaboration diagram for BookBaseEntry:



Public Member Functions

- **BookBaseEntry** ()
BookBaseEntry constructor.
- **BookBaseEntry** (const **BookBaseEntry** &other)
BookBaseEntry copy constructor.
- **BookBaseEntry** (**BookBaseEntry** &&other)
BookBaseEntry move constructor.
- **BookBaseEntry** & **operator=** (const **BookBaseEntry** &other)
operator =
- **BookBaseEntry** & **operator=** (**BookBaseEntry** &&other)
operator =
- bool **operator==** (const **BookBaseEntry** &other)
operator ==
- **BookBaseEntry** (const **BookParseEntry** &bpe, const std::filesystem::path &book_file_path)
BookBaseEntry constructor.

Public Attributes

- std::filesystem::path **file_path**
Absolute path to book file or archive.
- **BookParseEntry** **bpe**
BookParseEntry object.

4.8.1 Detailed Description

The **BookBaseEntry** class.

Auxiliary class, used to keep collections databases search requests and search results.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 BookBaseEntry()

```
BookBaseEntry::BookBaseEntry (
    const BookParseEntry & bpe,
    const std::filesystem::path & book_file_path)
```

BookBaseEntry constructor.

Parameters

<i>bpe</i>	BookParseEntry object.
<i>book_file_path</i>	absolute path to books file or archive.

4.9 BookInfo Class Reference

The [BookInfo](#) class.

```
#include <BookInfo.h>
```

Public Member Functions

- [BookInfo](#) (const std::shared_ptr< [AuxFunc](#) > &af)
BookInfo constructor.
- std::shared_ptr< [BookInfoEntry](#) > [get_book_info](#) (const [BookBaseEntry](#) &bbe)
Retruns information about book.
- void [set_dpi](#) (const double &h_dpi, const double &v_dpi)
Sets DPI.

4.9.1 Detailed Description

The [BookInfo](#) class.

This class contains methods to get extra information (like annotation, cover, source paper book info, etc) from books.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 BookInfo()

```
BookInfo::BookInfo (
    const std::shared_ptr< AuxFunc > & af)
```

[BookInfo](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.9.3 Member Function Documentation

4.9.3.1 get_book_info()

```
std::shared_ptr< BookInfoEntry > BookInfo::get_book_info (
    const BookBaseEntry & bbe)
```

Retruns information about book.

See also [set_dpi\(\)](#).

Parameters

<i>bbe</i>	search result, returned by BaseKeeper::searchBook() method.
------------	---

Returns

Smart pointer to [BookInfoEntry](#) object containing various information about book.

4.9.3.2 set_dpi()

```
void BookInfo::set_dpi (
    const double & h_dpi,
    const double & v_dpi)
```

Sets DPI.

This method should be called before [get_book_info\(\)](#). It sets **DPI** to display books cover correctly. Default values are 72.0 and 72.0. It is not compulsory to call this method, but it is highly recommended.

Parameters

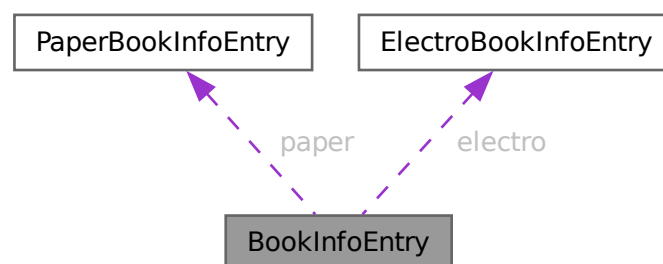
<i>h_dpi</i>	horizontal HREF="https://en.wikipedia.org/wiki/Dots_per_inch"> DPI .
<i>v_dpi</i>	vertical HREF="https://en.wikipedia.org/wiki/Dots_per_inch"> DPI .

4.10 BookInfoEntry Class Reference

The [BookInfoEntry](#) class.

```
#include <BookInfoEntry.h>
```

Collaboration diagram for BookInfoEntry:



Public Types

- enum [cover_types](#) {
[base64](#) , [file](#) , [rgb](#) , [rgba](#) ,
[bgra](#) , [text](#) , [error](#) }
The cover types enumerator.

Public Member Functions

- **BookInfoEntry** ()
[BookInfoEntry](#) constructor.
- virtual ~**BookInfoEntry** ()
[BookInfoEntry](#) destructor.
- **BookInfoEntry** (const [BookInfoEntry](#) &other)
[BookInfoEntry](#) copy constructor.
- **BookInfoEntry** ([BookInfoEntry](#) &&other)
[BookInfoEntry](#) move constructor.
- [BookInfoEntry](#) & **operator=** (const [BookInfoEntry](#) &other)
operator =
- [BookInfoEntry](#) & **operator=** ([BookInfoEntry](#) &&other)
operator =

Public Attributes

- std::string **annotation**
Book annotation.
- std::string **cover**
Book cover image.
- [cover_types](#) **cover_type** = [cover_types::error](#)
Type of image.
- std::string **language**
Book language.
- std::string **src_language**
Language of book source.
- std::string **translator**
Translator.
- [PaperBookInfoEntry](#) * **paper** = nullptr
Pointer to [PaperBookInfoEntry](#).
- [ElectroBookInfoEntry](#) * **electro** = nullptr
Various technical information about book file. See [ElectroBookInfoEntry](#).
- int **bytes_per_row** = 0
Number of bytes per row for RGB and RGBA images.

4.10.1 Detailed Description

The [BookInfoEntry](#) class.

Auxiliary class keeping various extra information about book (see [BookInfo](#)).

4.10.2 Member Enumeration Documentation

4.10.2.1 cover_types

Enumerator

base64	base64 image
file	various image files formats, like JPG
rgb	RGB image
rgba	RGBA image
bgra	BGRA image
text	Not image, just raw text
error	no image, default value

4.11 BookMarks Class Reference

The [BookMarks](#) class.

```
#include <BookMarks.h>
```

Public Member Functions

- [BookMarks](#) (const std::shared_ptr< [AuxFunc](#) > &af)
BookMarks constructor.
- virtual ~**BookMarks** ()
BookMarks destructor.
- int [createBookmark](#) (const std::string &col_name, const [BookBaseEntry](#) &bbe)
Creates bookmark.
- std::vector< std::tuple< std::string, [BookBaseEntry](#) > > [getBookMarks](#) ()
Returns bookmarks.
- void [removeBookmark](#) (const std::string &col_name, const [BookBaseEntry](#) &bbe)
Removes bookmark.

4.11.1 Detailed Description

The [BookMarks](#) class.

This class keeps and operates collection bookmarks. Path to bookmarks base is "~/local/share/MyLibrary/BookMarks/bookmarks".

4.11.2 Constructor & Destructor Documentation

4.11.2.1 BookMarks()

```
BookMarks::BookMarks (
    const std::shared_ptr< AuxFunc > & af)
```

[BookMarks](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.11.3 Member Function Documentation

4.11.3.1 createBookMark()

```
int BookMarks::createBookMark (
    const std::string & col_name,
    const BookBaseEntry & bbe)
```

Creates bookmark.

If bookmark already exists, returns 0, in case of success returns 1, otherwise returns -1.

Parameters

<i>col_name</i>	collection name book came from.
<i>bbe</i>	BookBaseEntry got from BaseKeeper::searchBook() .

Returns

Error code.

4.11.3.2 getBookMarks()

```
std::vector< std::tuple< std::string, BookBaseEntry > > BookMarks::getBookMarks ()
```

Returns bookmarks.

Returns

Vector of bookmarks tuples. First element of tuple is collection name, book came from. Second element is book entry.

4.11.3.3 removeBookMark()

```
void BookMarks::removeBookMark (
    const std::string & col_name,
    const BookBaseEntry & bbe)
```

Removes bookmark.

Parameters

<i>col_name</i>	collection name.
<i>bbe</i>	book entry to be removed.

4.12 BookParseEntry Class Reference

The [BookParseEntry](#) class.

```
#include <BookParseEntry.h>
```

Public Member Functions

- **BookParseEntry** ()
BookParseEntry constructor.
- **BookParseEntry** (const [BookParseEntry](#) &other)
BookParseEntry copy constructor.
- **BookParseEntry** ([BookParseEntry](#) &&other)
BookParseEntry move constructor.
- **BookParseEntry** & **operator=** (const [BookParseEntry](#) &other)
operator =
- **BookParseEntry** & **operator=** ([BookParseEntry](#) &&other)
operator =
- bool **operator==** (const [BookParseEntry](#) &other)
operator ==

Public Attributes

- std::string [book_path](#)
Path to book in file (in case of archive, empty otherwise).
- std::string **book_author**
Book author (if any, empty otherwise).
- std::string [book_name](#)
Book name.
- std::string **book_series**
Book series (if any, empty otherwise).
- std::string [book_genre](#)
Book genres(s) (if any, empty otherwise).
- std::string **book_date**
Book creation date (if available in metadata, empty otherwise).

4.12.1 Detailed Description

The [BookParseEntry](#) class.

Auxiliary class keeping relative path to book in file (in case of archive, empty otherwise), book author(s), book name, book series, book genre(s), date of book creation (if available).

4.12.2 Member Data Documentation

4.12.2.1 book_genre

```
std::string BookParseEntry::book_genre
```

Book genres(s) (if any, empty otherwise).

List of genres separated by ", " sequence.

4.12.2.2 book_name

```
std::string BookParseEntry::book_name
```

Book name.

Must not be empty. If book name cannot be obtained from book metadata, book file name will be used.

4.12.2.3 book_path

```
std::string BookParseEntry::book_path
```

Path to book in file (in case of archive, empty otherwise).

In case of "archive inside archive" situation "\n" (ASCII new line) symbol used as separator. It means that path to book inside archive looks like "<archive_one>\n<archive_two>\n<archive_three>\n<book_file>" or "<archive_one>\n<book_file>".

4.13 ByteOrder Class Reference

The [ByteOrder](#) class.

```
#include <ByteOrder.h>
```

Public Member Functions

- **ByteOrder** ()
ByteOrder constructor.
- virtual ~**ByteOrder** ()
ByteOrder destructor.
- **ByteOrder** (const [ByteOrder](#) &other)
ByteOrder copy constructor.
- [ByteOrder](#) (uint64_t val)
ByteOrder constructor.
- [ByteOrder](#) (uint32_t val)
ByteOrder constructor.
- [ByteOrder](#) (uint16_t val)
ByteOrder constructor.

- [ByteOrder](#) (int64_t val)
ByteOrder constructor.
- [ByteOrder](#) (int32_t val)
ByteOrder constructor.
- [ByteOrder](#) (int16_t val)
ByteOrder constructor.
- [ByteOrder](#) (float val)
ByteOrder constructor.
- [ByteOrder](#) (double val)
ByteOrder constructor.
- [ByteOrder](#) & **operator=** (const [ByteOrder](#) &other)
operator =
- [ByteOrder](#) & **operator=** (const uint64_t &val)
operator =
- [ByteOrder](#) & **operator=** (const uint32_t &val)
operator =
- [ByteOrder](#) & **operator=** (const uint16_t &val)
operator =
- [ByteOrder](#) & **operator=** (const int64_t &val)
operator =
- [ByteOrder](#) & **operator=** (const int32_t &val)
operator =
- [ByteOrder](#) & **operator=** (const int16_t &val)
operator =
- [ByteOrder](#) & **operator=** (const float &val)
operator =
- [ByteOrder](#) & **operator=** (const double &val)
operator =
- **operator uint64_t** ()
Returns number in native byte order.
- **operator uint32_t** ()
Returns number in native byte order.
- **operator uint16_t** ()
Returns number in native byte order.
- **operator int64_t** ()
Returns number in native byte order.
- **operator int32_t** ()
Returns number in native byte order.
- **operator int16_t** ()
Returns number in native byte order.
- **operator float** ()
Returns number in native byte order.
- **operator double** ()
Returns number in native byte order.
- template<typename T>
void [get_native](#) (T &result)
Returns number in native byte order.
- template<typename T>
void [get_big](#) (T &result)
Returns "big endian" number.

- `template<typename T>`
`void get_little (T &result)`
Returns "little endian" number.
- `template<typename T>`
`void set_big (T val)`
Sets inner value to val.
- `template<typename T>`
`void set_little (T val)`
Sets inner value to val.

4.13.1 Detailed Description

The [ByteOrder](#) class.

Auxiliary class. Is used to convert numbers to different byte orders.

Warning

If you get number from [ByteOrder](#), resulting number must have same type as input value, otherwise behavior is undefined.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 [ByteOrder\(\)](#) [1/8]

```
ByteOrder::ByteOrder (
    uint64_t val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.2.2 [ByteOrder\(\)](#) [2/8]

```
ByteOrder::ByteOrder (
    uint32_t val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.2.3 ByteOrder() [3/8]

```
ByteOrder::ByteOrder (  
    uint16_t val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.2.4 ByteOrder() [4/8]

```
ByteOrder::ByteOrder (  
    int64_t val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.2.5 ByteOrder() [5/8]

```
ByteOrder::ByteOrder (  
    int32_t val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.2.6 ByteOrder() [6/8]

```
ByteOrder::ByteOrder (  
    int16_t val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.2.7 ByteOrder() [7/8]

```
ByteOrder::ByteOrder (  
    float val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.2.8 ByteOrder() [8/8]

```
ByteOrder::ByteOrder (  
    double val)
```

[ByteOrder](#) constructor.

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3 Member Function Documentation**4.13.3.1 get_big()**

```
template<typename T>  
void ByteOrder::get_big (  
    T & result)
```

Returns "big endian" number.

Parameters

<i>result</i>	resulting number.
---------------	-------------------

4.13.3.2 get_little()

Parameters

<i>result</i>	resulting number.
---------------	-------------------

4.13.3.3 get_native()

```
template<typename T>
void ByteOrder::get_native (
    T & result)
```

Returns number in native byte order.

Parameters

<i>result</i>	resulting number.
---------------	-------------------

4.13.3.4 operator=() [1/8]

```
ByteOrder & ByteOrder::operator= (
    const double & val)
```

operator =

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.5 operator=() [2/8]

```
ByteOrder & ByteOrder::operator= (
    const float & val)
```

operator =

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.6 operator=() [3/8]

```
ByteOrder & ByteOrder::operator= (
    const int16_t & val)
```

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.7 operator=() [4/8]

```
ByteOrder & ByteOrder::operator= (  
    const int32_t & val)
```

operator =

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.8 operator=() [5/8]

```
ByteOrder & ByteOrder::operator= (  
    const int64_t & val)
```

operator =

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.9 operator=() [6/8]

```
ByteOrder & ByteOrder::operator= (  
    const uint16_t & val)
```

operator =

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.10 operator=() [7/8]

```
ByteOrder & ByteOrder::operator= (  
    const uint32_t & val)
```


Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.11 operator=() [8/8]

```
ByteOrder & ByteOrder::operator= (  
    const uint64_t & val)
```

operator =

Parameters

<i>val</i>	value in native byte order.
------------	-----------------------------

4.13.3.12 set_big()

```
template<typename T>  
void ByteOrder::set_big (  
    T val)
```

Sets inner value to val.

Sets inner value to val. val will be processed as "big endian".

Parameters

<i>val</i>	value to be set.
------------	------------------

4.13.3.13 set_little()

```
template<typename T>  
void ByteOrder::set_little (  
    T val)
```

Sets inner value to val.

Sets inner value to val. val will be processed as "little endian".

Parameters

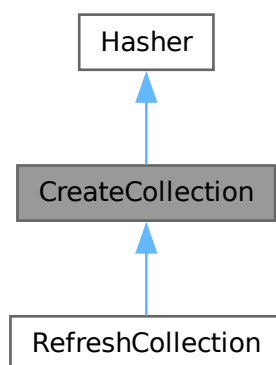
<i>val</i>	value to be set.
------------	------------------

4.14 CreateCollection Class Reference

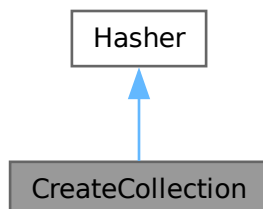
The [CreateCollection](#) class.

```
#include <CreateCollection.h>
```

Inheritance diagram for CreateCollection:



Collaboration diagram for CreateCollection:



Public Member Functions

- [CreateCollection](#) (const std::shared_ptr< [AuxFunc](#) > &af, const std::filesystem::path &collection_path, const std::filesystem::path &books_path, const bool &rar_support, const int &num_threads)
CreateCollection constructor.
- virtual ~**CreateCollection** ()
CreateCollection destructor.
- void [createCollection](#) ()
Starts collection creation.

Public Member Functions inherited from [Hasher](#)

- [Hasher](#) (const std::shared_ptr< [AuxFunc](#) > &af)
Hasher constructor.
- std::string [buf_hashing](#) (const std::string &buf)
Creates hash sum for given buffer.
- std::string [file_hashing](#) (const std::filesystem::path &filepath)
Creates hash sum for given file.
- void **cancelAll** ()
Stops all operations.

Public Attributes

- std::function< void()> [pulse](#)
"Pulse" callback.
- std::function< void(const double &)> [signal_total_bytes](#)
"Total bytes" callback.
- std::function< void(const double &progress)> [progress](#)
"Progress" callback.

Protected Member Functions

- [CreateCollection](#) (const std::shared_ptr< [AuxFunc](#) > &af, const int &num_threads)
CreateCollection constructor.
- void [threadRegulator](#) ()
Threads regulator.
- void [openBaseFile](#) ()
Opens database file for writing.
- void [closeBaseFile](#) ()
Finishes database writing.
- void [write_file_to_base](#) (const [FileParseEntry](#) &fe)
Writes file data to database.

Protected Attributes

- `std::filesystem::path` [base_path](#)
Absolute path to database.
- `std::filesystem::path` [books_path](#)
Absolute path to books directory.
- `bool` [rar_support](#) = false
If true, rar archives will be processed, otherwise - not.
- `std::vector< std::tuple< std::filesystem::path, std::string > >` [already_hashed](#)
Hashed files.
- `std::vector< std::filesystem::path >` [need_to_parse](#)
"Need to parse" vector.
- `std::atomic< double >` [current_bytes](#)
Keeps quantity of bytes have been processed.

Protected Attributes inherited from [Hasher](#)

- `std::atomic< bool >` [cancel](#)
Stops all operations if true.
- `std::function< void()>` [stop_all_signal](#)
Stop signal for heir classes.

4.14.1 Detailed Description

The [CreateCollection](#) class.

This class contains methods for collection database creation.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 [CreateCollection\(\)](#) [1/2]

```
CreateCollection::CreateCollection (
    const std::shared_ptr< AuxFunc > & af,
    const std::filesystem::path & collection_path,
    const std::filesystem::path & books_path,
    const bool & rar_support,
    const int & num_threads)
```

[CreateCollection](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
<i>collection_path</i>	absolute path to collection base directory (if it doesnt exist, MLBookProc will create it).
<i>books_path</i>	absolute path to books directory.

<i>rar_support</i>	if <i>true</i> , rar archives will be processed, otherwise - not (some rar archives can cause errors).
<i>num_threads</i>	limit of working threads to be used.

4.14.2.2 CreateCollection() [2/2]

```
CreateCollection::CreateCollection (
    const std::shared_ptr< AuxFunc > & af,
    const int & num_threads) [protected]
```

[CreateCollection](#) constructor.

Warning

Do not call this constructor yourself!

This constructor is used by [RefreshCollection](#) class.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
<i>num_threads</i>	limit of working threads to be used.

4.14.3 Member Function Documentation

4.14.3.1 closeBaseFile()

```
void CreateCollection::closeBaseFile () [protected]
```

Finishes database writing.

Warning

Do not call this method yourself!

4.14.3.2 createCollection()

```
void CreateCollection::createCollection ()
```

Starts collection creation.

Note

This method can throw [MLException](#) in case of errors.

4.14.3.3 openBaseFile()

```
void CreateCollection::openBaseFile () [protected]
```

Opens database file for writing.

Warning

Do not call this method yourself!

4.14.3.4 threadRegulator()

```
void CreateCollection::threadRegulator () [protected]
```

Threads regulator.

Warning

Do not call this method yourself!

4.14.3.5 write_file_to_base()

```
void CreateCollection::write_file_to_base (  
    const FileParseEntry & fe) [protected]
```

Writes file data to database.

Warning

Do not call this method yourself!

Parameters

<i>fe</i>	FileParseEntry object.
-----------	--

4.14.4 Member Data Documentation

4.14.4.1 already_hashed

```
std::vector<std::tuple<std::filesystem::path, std::string> > CreateCollection::already_hashed  
[protected]
```

Hashed files.

This vector is used by [RefreshCollection](#) class to indicate files, has been already hashed.

Warning

Do not call or set this vector yourself!

4.14.4.2 base_path

```
std::filesystem::path CreateCollection::base_path [protected]
```

Absolute path to database.

Warning

Do not call or set this variable yourself!

4.14.4.3 books_path

```
std::filesystem::path CreateCollection::books_path [protected]
```

Absolute path to books directory.

Warning

Do not call or set this variable yourself!

4.14.4.4 current_bytes

```
std::atomic<double> CreateCollection::current_bytes [protected]
```

Keeps quantity of bytes have been processed.

Warning

Do not call or set this variable yourself!

4.14.4.5 need_to_parse

```
std::vector<std::filesystem::path> CreateCollection::need_to_parse [protected]
```

"Need to parse" vector.

This vector is used to indicate files, needed to be parsed.

Warning

Do not call or set this vector yourself!

4.14.4.6 progress

```
std::function<void(const double &progress)> CreateCollection::progress
```

"Progress" callback.

Emitted after each file processing completion. Indicates total quantity of bytes has been processed. Bind your method to it, if you need such information.

4.14.4.7 pulse

```
std::function<void()> CreateCollection::pulse
```

"Pulse" callback.

Emitted while preliminary files collecting to show that process is not frozen. Bind your method to it, if you need such information.

4.14.4.8 rar_support

```
bool CreateCollection::rar_support = false [protected]
```

If *true*, rar archives will be processed, otherwise - not.

Warning

Do not call or set this variable yourself!

4.14.4.9 signal_total_bytes

```
std::function<void(const double &)> CreateCollection::signal_total_bytes
```

"Total bytes" callback.

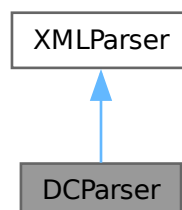
Emitted after preliminary files collecting completed to indicate total quantity of bytes to be processed. Bind your method to it, if you need such information.

4.15 DCParse Class Reference

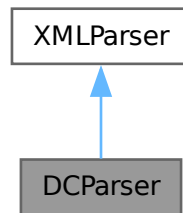
The [DCParser](#) class.

```
#include <DCParser.h>
```

Inheritance diagram for DCParse:



Collaboration diagram for DCParse:



Public Member Functions

- `DCParse` (const std::shared_ptr< [AuxFunc](#) > &af)
DCParse constructor.
- std::string `dcTitle` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book title.
- std::string `dcAuthor` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book author.
- std::string `dcGenre` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book genre.
- std::string `dcDate` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book creation date.
- std::string `dcLanguage` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book language.
- std::string `dcPublisher` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets file publisher.
- std::string `dcIdentifier` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book identifier.
- std::string `dcSource` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book source.
- std::string `dcDescription` (const std::string &dc_file_content, const std::vector< [XMLTag](#) > &tgvs)
Gets book description.

Public Member Functions inherited from [XMLParser](#)

- `XMLParser` (const std::shared_ptr< [AuxFunc](#) > &af)
XMLParser constructor.
- std::vector< [XMLTag](#) > `get_tag` (const std::string &book, const std::string &tag_id)
Returns all tags with particular name.
- std::string `get_book_encoding` (const std::string &book)
Returns [XML](#) document encoding.
- std::string `get_element_attribute` (const std::string &element, const std::string &attr_name)
Returns [XML](#) tag attribute if it was found.
- std::vector< [XMLTag](#) > `listAllTags` (const std::string &book)

Parses *XML* document.

- void [searchTag](#) (const std::vector< [XMLTag](#) > &list, const std::string &tag_id, std::vector< [XMLTag](#) > &result)

Searches tag in tag list.

- void [htmlSymbolsReplacement](#) (std::string &book)

Replaces symbols encoded by "&..." sequences.

- void [removeAllTags](#) (std::string &book)

Removes all tag elements from *XML* document.

4.15.1 Detailed Description

The [DCParser](#) class.

Auxiliary class. Contains methods for *DublinCore* files parsing. This class is used in [ODTParser](#) and [EPUBParser](#). You do not need to call this class methods directly.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 DCParser()

```
DCParser::DCParser (
    const std::shared_ptr< AuxFunc > & af)
```

[DCParser](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.15.3 Member Function Documentation

4.15.3.1 dcAuthor()

```
std::string DCParser::dcAuthor (
    const std::string & dc_file_content,
    const std::vector< XMLTag > & tgv)
```

Gets book author.

This method can be used to get author(s) from *DublinCore* file.

Parameters

<i>dc_file_content</i>	string containing file content.
------------------------	---------------------------------

<i>tg</i>	XMLTag vector obtained from XMLParser::listAllTags method.
-----------	--

Returns

Author if any, empty string otherwise.

4.15.3.2 dcDate()

```
std::string DCParse::dcDate (  
    const std::string & dc_file_content,  
    const std::vector< XMLTag > & tg
```

Gets book creation date.

This method can be used to get creation date from [DublinCore](#) file.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tg</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

File creation date if any, empty otherwise.

4.15.3.3 dcDescription()

```
std::string DCParse::dcDescription (  
    const std::string & dc_file_content,  
    const std::vector< XMLTag > & tg
```

Gets book description.

This method can be used to get book description from [DublinCore](#) file.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tg</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

Generated by Doxygen

Book description if any, empty otherwise.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tgvs</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

Genre if any, empty string otherwise.

4.15.3.5 dcIdentifier()

```
std::string DCParser::dcIdentifier (
    const std::string & dc_file_content,
    const std::vector< XMLTag > & tgvs)
```

Gets book identifier.

This method can be used to get identifier from [DublinCore](#) file.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tgvs</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

Book identifier if any, empty otherwise.

4.15.3.6 dcLanguage()

```
std::string DCParse::dcLanguage (
    const std::string & dc_file_content,
    const std::vector< XMLTag > & tgvs)
```

Gets book language.

This method can be used to get language from [DublinCore](#) file.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tgvs</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

File language if set, empty otherwise.

4.15.3.7 dcPublisher()

```
std::string DCParser::dcPublisher (
    const std::string & dc_file_content,
    const std::vector< XMLTag > & tgv)
```

Gets file publisher.

This method can be used to get publisher from **DublinCore** file.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tgv</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

File publisher if set, empty otherwise.

4.15.3.8 dcSource()

```
std::string DCParser::dcSource (
    const std::string & dc_file_content,
    const std::vector< XMLTag > & tgv)
```

Gets book source.

This method can be used to get book source from **DublinCore** file.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tgv</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

Book source if set, empty otherwise.

4.15.3.9 dcTitle()

```
std::string DCParser::dcTitle (
    const std::string & dc_file_content,
    const std::vector< XMLTag > & tgv)
```

Generated by Doxygen
Gets book title.

This method can be used to get title from **DublinCore** file.

Parameters

<i>dc_file_content</i>	string containing file content.
<i>tgvs</i>	XMLTag vector obtained from XMLParser::listAllTags method.

Returns

Title if any, empty string otherwise.

4.16 DJVUParser Class Reference

The [DJVUParser](#) class.

```
#include <DJVUParser.h>
```

Public Member Functions

- [DJVUParser](#) (const std::shared_ptr< [AuxFunc](#) > &af)
DJVUParser constructor.
- [BookParseEntry djvu_parser](#) (const std::filesystem::path &filepath)
Parses djvu book.
- std::shared_ptr< [BookInfoEntry](#) > [djvu_book_info](#) (const std::filesystem::path &filepath)
Returns book info and book cover.

4.16.1 Detailed Description

The [DJVUParser](#) class.

This class contains various methods for djvu books processing. In most cases you do not need to use this class directly. Use [CreateCollection](#), [RefreshCollection](#) and [BookInfo](#) instead.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 DJVUParser()

```
DJVUParser::DJVUParser (
    const std::shared_ptr< AuxFunc > & af)
```

[DJVUParser](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.16.3 Member Function Documentation

Parameters

<i>filepath</i>	absolute path to djvu book.
-----------------	-----------------------------

Returns

Smart pointer to [BookInfoEntry](#) object.

4.16.3.2 djvu_parser()

```
BookParseEntry DJVUParser::djvu_parser (
    const std::filesystem::path & filepath)
```

Parses djvu book.

Parameters

<i>filepath</i>	absolute path to djvu book.
-----------------	-----------------------------

Returns

[BookParseEntry](#) object.

4.17 ElectroBookInfoEntry Class Reference

The [ElectroBookInfoEntry](#) class.

```
#include <ElectroBookInfoEntry.h>
```

Public Member Functions

- **ElectroBookInfoEntry ()**
ElectroBookInfoEntry constructor.
- **ElectroBookInfoEntry (const [ElectroBookInfoEntry](#) &other)**
ElectroBookInfoEntry copy constructor.
- **ElectroBookInfoEntry ([ElectroBookInfoEntry](#) &&other)**
ElectroBookInfoEntry move constructor.
- **[ElectroBookInfoEntry](#) & operator= (const [ElectroBookInfoEntry](#) &other)**
operator =
- **[ElectroBookInfoEntry](#) & operator= ([ElectroBookInfoEntry](#) &&other)**
operator =

Public Attributes

- bool **available** = false
Indicates if any information is available.
- std::string **author**
Author of file.
- std::string **program_used**
Program used to create file.
- std::string **date**
Date of file creation.
- std::string **src_url**
Source URL if this document is a conversion of some other (online) document.
- std::string **src_ocr**
Author of the original (online) document, if this is a conversion.
- std::string **id**
Unique file identifier.
- std::string **version**
File version.
- std::string **history**
History of file changes.
- std::string **publisher**
File publisher.

4.17.1 Detailed Description

The [ElectroBookInfoEntry](#) class.

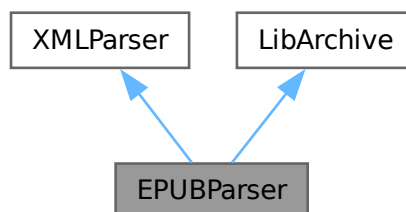
Auxiliary class containing various technical info about digital book file.

4.18 EPUBParser Class Reference

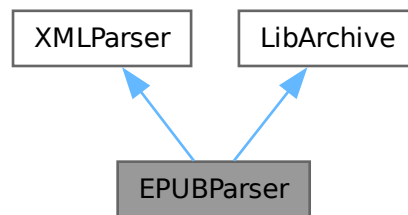
The [EPUBParser](#) class.

```
#include <EPUBParser.h>
```

Inheritance diagram for EPUBParser:



Collaboration diagram for EPUBParser:



Public Member Functions

- **EPUBParser** (const std::shared_ptr< [AuxFunc](#) > &af)
EPUBParser constructor.
- virtual ~**EPUBParser** ()
EPUBParser destructor.
- **BookParseEntry** [epub_parser](#) (const std::filesystem::path &filepath)
Parses epub book.
- std::shared_ptr< [BookInfoEntry](#) > [epub_book_info](#) (const std::filesystem::path &filepath)
Returns epub book info and cover.

Public Member Functions inherited from [XMLParser](#)

- **XMLParser** (const std::shared_ptr< [AuxFunc](#) > &af)
XMLParser constructor.
- std::vector< [XMLTag](#) > [get_tag](#) (const std::string &book, const std::string &tag_id)
Returns all tags with particular name.
- std::string [get_book_encoding](#) (const std::string &book)
Returns [XML](#) document encoding.
- std::string [get_element_attribute](#) (const std::string &element, const std::string &attr_name)
Returns [XML](#) tag attribute if it was found.
- std::vector< [XMLTag](#) > [listAllTags](#) (const std::string &book)
Parses [XML](#) document.
- void [searchTag](#) (const std::vector< [XMLTag](#) > &list, const std::string &tag_id, std::vector< [XMLTag](#) > &result)
Searches tag in tag list.
- void [htmlSymbolsReplacement](#) (std::string &book)
Replaces symbols encoded by "&..." sequences.
- void [removeAllTags](#) (std::string &book)
Removes all tag elements from [XML](#) document.

Public Member Functions inherited from [LibArchive](#)

- [LibArchive](#) (const std::shared_ptr< [AuxFunc](#) > &af)
LibArchive constructor.
- std::filesystem::path [unpackByPosition](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::string [unpackByPositionStr](#) (const std::filesystem::path &archaddress, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::filesystem::path [unpackByFileNameStream](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const std::string &filename)
Unpacks entry content from archive.
- std::string [unpackByFileNameStreamStr](#) (const std::filesystem::path &archaddress, const std::string &filename)
Unpacks entry content from archive.
- int [fileNames](#) (const std::filesystem::path &filepath, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.
- int [fileNamesStream](#) (const std::filesystem::path &address, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.
- [ArchEntry](#) [fileinfo](#) (const std::filesystem::path &address, const std::string &filename)
Returns ArchEntry for particular file or directory in archive.
- int [libarchive_packing](#) (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)
Packs file or directory into archive.
- int [libarchive_packing](#) (const std::shared_ptr< archive > &a, const std::filesystem::path &sourcepath, const bool &rename_source, const std::string &new_source_name)
Packs file or directory into archive.
- [ArchiveRemoveEntry](#) [libarchive_remove_init](#) (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)
Initializes archive objects for removing entries from archive.
- int [libarchive_remove_entry](#) ([ArchiveRemoveEntry](#) rm_e, const std::vector< [ArchEntry](#) > &to_remove)
Removes entry from archive.
- void [libarchive_error](#) (const std::shared_ptr< archive > &a, const std::string &message, const int &error_number)
Prints libarchive error messages.
- std::string [libarchive_read_entry_str](#) (archive *a, archive_entry *entry)
Reads archived file to string.
- int [libarchive_write_data](#) (archive *a, const std::string &data)
Writes data to archive.
- std::shared_ptr< [ArchiveFileEntry](#) > [createArchFile](#) (const std::filesystem::path &archaddress, const int64_t &position=la_int64_t(0))
Creates ArchiveFileEntry object.
- std::shared_ptr< archive > [libarchive_read_init](#) (std::shared_ptr< [ArchiveFileEntry](#) > fl)
Initializes archive reading.
- std::shared_ptr< archive > [libarchive_read_init_fallback](#) (std::shared_ptr< [ArchiveFileEntry](#) > fl)
Initializes archive reading.
- std::filesystem::path [libarchive_read_entry](#) (archive *a, archive_entry *entry, const std::filesystem::path &outfolder)
Unpacks libarchive entry content.
- std::shared_ptr< archive > [libarchive_write_init](#) (const std::filesystem::path &outpath)
Initializes writing to archive.
- int [writeToArchive](#) (std::shared_ptr< archive > a, const std::filesystem::path &source, const std::filesystem::path &path_in_arch)

Writes file or directory to archive.

- int [libarchive_write_directory](#) (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch, const std::filesystem::path &source)

Writes directory to archive.

- int [libarchive_write_file](#) (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch, const std::filesystem::path &source)

Writes file to archive.

- int [libarchive_write_data_from_file](#) (archive *a, const std::filesystem::path &source)

Writes raw data from file to archive.

4.18.1 Detailed Description

The [EPUBParser](#) class.

This class contains various methods for epub books parsing. In most cases you do not need to use this class directly. Use [CreateCollection](#), [RefreshCollection](#) and [BookInfo](#) instead.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 EPUBParser()

```
EPUBParser::EPUBParser (
    const std::shared_ptr< AuxFunc > & af)
```

[EPUBParser](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.18.3 Member Function Documentation

4.18.3.1 epub_book_info()

```
std::shared_ptr< BookInfoEntry > EPUBParser::epub_book_info (
    const std::filesystem::path & filepath)
```

Returns epub book info and cover.

Parameters

<i>filepath</i>	absolute path to epub file.
-----------------	-----------------------------

Parameters

<i>filepath</i>	absolute path to epub file.
-----------------	-----------------------------

Returns

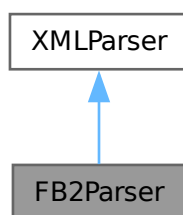
[BookParseEntry](#) object.

4.19 FB2Parser Class Reference

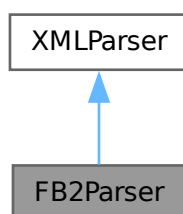
The [FB2Parser](#) class.

```
#include <FB2Parser.h>
```

Inheritance diagram for FB2Parser:



Collaboration diagram for FB2Parser:



Public Member Functions

- [FB2Parser](#) (const std::shared_ptr< [AuxFunc](#) > &af)
FB2Parser constructor.
- [BookParseEntry fb2_parser](#) (const std::string &book)
Parses fb2 book.
- std::shared_ptr< [BookInfoEntry](#) > [fb2_book_info](#) (const std::string &book)
Returns fb2 book info and cover.

Public Member Functions inherited from [XMLParser](#)

- [XMLParser](#) (const std::shared_ptr< [AuxFunc](#) > &af)
XMLParser constructor.
- std::vector< [XMLTag](#) > [get_tag](#) (const std::string &book, const std::string &tag_id)
Returns all tags with particular name.
- std::string [get_book_encoding](#) (const std::string &book)
Returns XML document encoding.
- std::string [get_element_attribute](#) (const std::string &element, const std::string &attr_name)
Returns XML tag attribute if it was found.
- std::vector< [XMLTag](#) > [listAllTags](#) (const std::string &book)
Parses XML document.
- void [searchTag](#) (const std::vector< [XMLTag](#) > &list, const std::string &tag_id, std::vector< [XMLTag](#) > &result)
Searches tag in tag list.
- void [htmlSymbolsReplacement](#) (std::string &book)
Replaces symbols encoded by "&..." sequences.
- void [removeAllTags](#) (std::string &book)
Removes all tag elements from XML document.

4.19.1 Detailed Description

The [FB2Parser](#) class.

This class contains various methods for fb2 books parsing. In most cases you do not need to use this class directly. Use [CreateCollection](#), [RefreshCollection](#) and [BookInfo](#) instead.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 FB2Parser()

```
FB2Parser::FB2Parser (
    const std::shared_ptr< AuxFunc > & af)
```

[FB2Parser](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.19.3 Member Function Documentation

4.19.3.1 fb2_book_info()

```
std::shared_ptr< BookInfoEntry > FB2Parser::fb2_book_info (  
    const std::string & book)
```

Returns fb2 book info and cover.

Parameters

<i>book</i>	fb2 books file content.
-------------	-------------------------

Returns

Smart pointer to [BookInfoEntry](#) object.

4.19.3.2 fb2_parser()

```
BookParseEntry FB2Parser::fb2_parser (  
    const std::string & book)
```

Parses fb2 book.

Note

This method can throw [MLException](#) object in case of any errors.

Parameters

<i>book</i>	fb2 file content.
-------------	-------------------

Returns

[BookParseEntry](#) object.

4.20 FileParseEntry Class Reference

The [FileParseEntry](#) class.

```
#include <FileParseEntry.h>
```

Public Member Functions

- **FileParseEntry** ()
FileParseEntry constructor.
- **FileParseEntry** (const [FileParseEntry](#) &other)
FileParseEntry copy constructor.
- **FileParseEntry** ([FileParseEntry](#) &&other)
FileParseEntry move constructor.
- **FileParseEntry** & **operator=** (const [FileParseEntry](#) &other)
operator =
- **FileParseEntry** & **operator=** ([FileParseEntry](#) &&other)
operator =

Public Attributes

- std::string [file_rel_path](#)
Relative path to file in collection.
- std::string [file_hash](#)
File hash sum.
- std::vector< [BookParseEntry](#) > [books](#)
Contains books info.

4.20.1 Detailed Description

The [FileParseEntry](#) class.

Auxiliary class, used in [CreateCollection](#), [RefreshCollection](#) and [BaseKeeper](#). Contains collection file info.

4.20.2 Member Data Documentation

4.20.2.1 books

```
std::vector<BookParseEntry> FileParseEntry::books
```

Contains books info.

Vector containing information about books in file.

4.20.2.2 file_hash

```
std::string FileParseEntry::file_hash
```

File hash sum.

Blake-256 algorithm currently used.

4.20.2.3 file_rel_path

```
std::string FileParseEntry::file_rel_path
```

Relative path to file in collection.

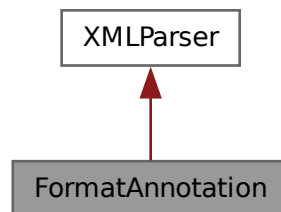
Path is relative to collection directory path.

4.21 FormatAnnotation Class Reference

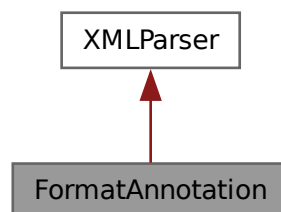
The [FormatAnnotation](#) class.

```
#include <FormatAnnotation.h>
```

Inheritance diagram for FormatAnnotation:



Collaboration diagram for FormatAnnotation:



Public Member Functions

- [FormatAnnotation](#) (const std::shared_ptr< [AuxFunc](#) > &af)
FormatAnnotation constructor.
- void [remove_escape_sequences](#) (std::string &annotation)
Removes escape sequences from annotation.
- void [replace_tags](#) (std::string &annotation)
Replaces XML tags.
- void [final_cleaning](#) (std::string &annotation)
Cleans some sequences from annotation.
- void [removeAllTags](#) (std::string &annotation)
Simply removes all XML tags.
- void [setTagReplacementTable](#) (const std::vector< [ReplaceTagItem](#) > &replacement_table)
Sets tag replacement table.

4.21.1 Detailed Description

The [FormatAnnotation](#) class.

This class contains different methods for annotation processing.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 FormatAnnotation()

```
FormatAnnotation::FormatAnnotation (
    const std::shared_ptr< AuxFunc > & af)
```

[FormatAnnotation](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.21.3 Member Function Documentation

4.21.3.1 final_cleaning()

```
void FormatAnnotation::final_cleaning (
    std::string & annotation)
```

Cleans some sequences from annotation.

Replaces "three spaces" by "two spaces" at the annotation beginning, removes "\n" from annotation beginning, removes 0 - 32 [ASCII](#) symbols from the annotation end. Also replaces "\n\n\n" sequences by "\n\n" in whole annotation.

Parameters

<i>annotation</i>	UTF-8 annotation content string.
-------------------	----------------------------------

4.21.3.2 `remove_escape_sequences()`

```
void FormatAnnotation::remove_escape_sequences (
    std::string & annotation)
```

Removes escape sequences from annotation.

Removes [ASCII](#) symbols 0 to 31 (inclusive) from annotation. Only exception is symbol 9 (horizontal tab). It will be replaced by 32 (space). Also removes extra spaces from annotation beginning.

Parameters

<i>annotation</i>	UTF-8 annotation content string.
-------------------	----------------------------------

4.21.3.3 `removeAllTags()`

```
void FormatAnnotation::removeAllTags (
    std::string & annotation)
```

Simply removes all XML tags.

Parameters

<i>annotation</i>	UTF-8 annotation content string.
-------------------	----------------------------------

4.21.3.4 `replace_tags()`

```
void FormatAnnotation::replace_tags (
    std::string & annotation)
```

Replaces XML tags.

It is highly recommended to call [setTagReplacementTable\(\)](#) method before this method call (but it is not compulsory). If tag replacement table is empty, all tags will be just removed.

Parameters

<i>annotation</i>	UTF-8 annotation content string.
-------------------	----------------------------------

4.21.3.5 setTagReplacementTable()

```
void FormatAnnotation::setTagReplacementTable (
    const std::vector< ReplaceTagItem > & replacement_table)
```

Sets tag replacement table.

If you need to replace XML tags by your own tags or text, you should call this method before [replace_tags\(\)](#) call to set replacement table.

Parameters

<i>replacement_table</i>	vector containing ReplaceTagItem objects.
--------------------------	---

4.22 Genre Class Reference

The [Genre](#) class.

```
#include <Genre.h>
```

Public Member Functions

- **Genre ()**
Genre constructor.
- **Genre (const Genre &other)**
Genre copy constructor.
- **Genre (Genre &&other)**
Genre move constructor.
- **Genre & operator= (const Genre &other)**
operator =
- **Genre & operator= (Genre &&other)**
operator =

Public Attributes

- std::string [genre_code](#)
fb2 genre code.
- std::string [genre_name](#)
Translated human-readable genre name.

4.22.1 Detailed Description

The [Genre](#) class.

Auxiliary class containing translated genre element (see also [AuxFunc::get_genre_list\(\)](#) and [GenreGroup](#)).

4.22.2 Member Data Documentation

4.22.2.1 genre_code

```
std::string Genre::genre_code
```

fb2 genre code.

This code is usually valid for epub books also.

4.22.2.2 genre_name

```
std::string Genre::genre_name
```

Translated human-readable genre name.

If translation is not available, English genre name will be set. For translations see "<share_path>/MLBookProc/genres.csv"

4.23 GenreGroup Class Reference

The [GenreGroup](#) class.

```
#include <GenreGroup.h>
```

Public Member Functions

- **GenreGroup** ()
GenreGroup constructor.
- **GenreGroup** (const [GenreGroup](#) &other)
GenreGroup copy constructor.
- **GenreGroup** ([GenreGroup](#) &&other)
GenreGroup move constructor.
- [GenreGroup](#) & **operator=** (const [GenreGroup](#) &other)
operator =
- [GenreGroup](#) & **operator=** ([GenreGroup](#) &&other)
operator =

Public Attributes

- `std::string group_code`
Genre group code name.
- `std::string group_name`
Translated human-readable genre group name.
- `std::vector< Genre > genres`
List of group genres (see Genre).

4.23.1 Detailed Description

The [GenreGroup](#) class.

Auxiliary class containing genre group (see also [AuxFunc::get_genre_list\(\)](#)).

4.23.2 Member Data Documentation**4.23.2.1 group_name**

```
std::string GenreGroup::group_name
```

Translated human-readable genre group name.

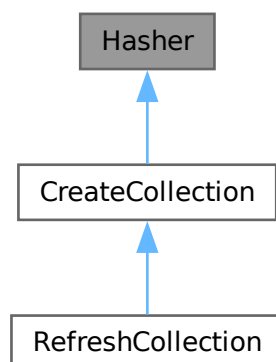
If translation is not available, English genre group name will be set. For translations see "<share_path>/MLBook↔Proc/genre_groups.csv"

4.24 Hasher Class Reference

The [Hasher](#) class.

```
#include <Hasher.h>
```

Inheritance diagram for Hasher:



Public Member Functions

- [Hasher](#) (const std::shared_ptr< [AuxFunc](#) > &af)
Hasher constructor.
- std::string [buf_hashing](#) (const std::string &buf)
Creates hash sum for given buffer.
- std::string [file_hashing](#) (const std::filesystem::path &filepath)
Creates hash sum for given file.
- void **cancelAll** ()
Stops all operations.

Protected Attributes

- std::atomic< bool > [cancel](#)
Stops all operations if true.
- std::function< void()> [stop_all_signal](#)
Stop signal for heir classes.

4.24.1 Detailed Description

The [Hasher](#) class.

This class contains methods for hash sums creating (Blake-256 algorithm).

4.24.2 Constructor & Destructor Documentation

4.24.2.1 Hasher()

```
Hasher::Hasher (
    const std::shared_ptr< AuxFunc > & af)
```

[Hasher](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.24.3 Member Function Documentation

4.24.3.1 buf_hashing()

```
std::string Hasher::buf_hashing (
    const std::string & buf)
```

Creates hash sum for given buffer.

Note

This method can throw [MLException](#) in case of error.

Parameters

<i>buf</i>	source buffer.
------------	----------------

Returns

32 bytes of hash sum value.

4.24.3.2 file_hashing()

```
std::string Hasher::file_hashing (  
    const std::filesystem::path & filepath)
```

Creates hash sum for given file.

Note

This method can throw [MLException](#) in case of error.

Parameters

<i>filepath</i>	absolute path to file to be hashed.
-----------------	-------------------------------------

Returns

32 bytes of hash sum value.

4.24.4 Member Data Documentation

4.24.4.1 cancel

```
std::atomic<bool> Hasher::cancel [protected]
```

Stops all operations if *true*.

Warning

Do not call or set this variable yourself!

4.24.4.2 stop_all_signal

```
std::function<void()> Hasher::stop_all_signal [protected]
```

Stop signal for heir classes.

Warning

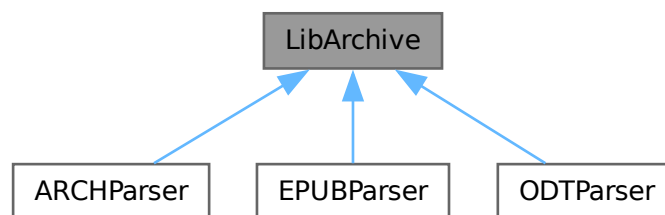
Do not call or set this variable yourself!

4.25 LibArchive Class Reference

The [LibArchive](#) class.

```
#include <LibArchive.h>
```

Inheritance diagram for LibArchive:



Public Member Functions

- [LibArchive](#) (const std::shared_ptr< [AuxFunc](#) > &af)
LibArchive constructor.
- std::filesystem::path [unpackByPosition](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::string [unpackByPositionStr](#) (const std::filesystem::path &archaddress, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::filesystem::path [unpackByFileNameStream](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const std::string &filename)
Unpacks entry content from archive.
- std::string [unpackByFileNameStreamStr](#) (const std::filesystem::path &archaddress, const std::string &filename)
Unpacks entry content from archive.
- int [fileNames](#) (const std::filesystem::path &filepath, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.
- int [fileNamesStream](#) (const std::filesystem::path &address, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.

- [ArchEntry](#) [fileinfo](#) (const std::filesystem::path &address, const std::string &filename)
Returns [ArchEntry](#) for particular file or directory in archive.
- int [libarchive_packing](#) (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)
Packs file or directory into archive.
- int [libarchive_packing](#) (const std::shared_ptr< archive > &a, const std::filesystem::path &sourcepath, const bool &rename_source, const std::string &new_source_name)
Packs file or directory into archive.
- [ArchiveRemoveEntry](#) [libarchive_remove_init](#) (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)
Initializes archive objects for removing entries from archive.
- int [libarchive_remove_entry](#) ([ArchiveRemoveEntry](#) rm_e, const std::vector< [ArchEntry](#) > &to_remove)
Removes entry from archive.
- void [libarchive_error](#) (const std::shared_ptr< archive > &a, const std::string &message, const int &error_number)
Prints libarchive error messages.
- std::string [libarchive_read_entry_str](#) (archive *a, archive_entry *entry)
Reads archived file to string.
- int [libarchive_write_data](#) (archive *a, const std::string &data)
Writes data to archive.
- std::shared_ptr< [ArchiveFileEntry](#) > [createArchFile](#) (const std::filesystem::path &archaddress, const la_int64_t &position=la_int64_t(0))
Creates [ArchiveFileEntry](#) object.
- std::shared_ptr< archive > [libarchive_read_init](#) (std::shared_ptr< [ArchiveFileEntry](#) > fl)
Initializes archive reading.
- std::shared_ptr< archive > [libarchive_read_init_fallback](#) (std::shared_ptr< [ArchiveFileEntry](#) > fl)
Initializes archive reading.
- std::filesystem::path [libarchive_read_entry](#) (archive *a, archive_entry *entry, const std::filesystem::path &out_folder)
Unpacks libarchive entry content.
- std::shared_ptr< archive > [libarchive_write_init](#) (const std::filesystem::path &outpath)
Initializes writing to archive.
- int [writeToArchive](#) (std::shared_ptr< archive > a, const std::filesystem::path &source, const std::filesystem::path &path_in_arch)
Writes file or directory to archive.
- int [libarchive_write_directory](#) (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch, const std::filesystem::path &source)
Writes directory to archive.
- int [libarchive_write_file](#) (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch, const std::filesystem::path &source)
Writes file to archive.
- int [libarchive_write_data_from_file](#) (archive *a, const std::filesystem::path &source)
Writes raw data from file to archive.

4.25.1 Detailed Description

The [LibArchive](#) class.

This class contains various methods for archives processing. Based on [libarchive](#) library.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 LibArchive()

```
LibArchive::LibArchive (
    const std::shared_ptr< AuxFunc > & af)
```

[LibArchive](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.25.3 Member Function Documentation

4.25.3.1 createArchFile()

```
std::shared_ptr< ArchiveFileEntry > LibArchive::createArchFile (
    const std::filesystem::path & archaddress,
    const la_int64_t & position = la_int64_t(0))
```

Creates [ArchiveFileEntry](#) object.

[ArchiveFileEntry](#) object is used in [libarchive_read_init\(\)](#) and [libarchive_read_init_fallback\(\)](#) methods.

Parameters

<i>archaddress</i>	absolute path to archive to be read.
<i>position</i>	position in archive file to start reading from.

Returns

Smart pointer to [ArchiveFileEntry](#) object.

4.25.3.2 fileinfo()

```
ArchEntry LibArchive::fileinfo (
    const std::filesystem::path & address,
    const std::string & filename)
```

Returns [ArchEntry](#) for particular file or directory in archive.

In case of any error [ArchEntry](#) filename will be empty.

Parameters

<i>address</i>	absolute path to archive.
<i>filename</i>	path in archive.

Returns

[ArchEntry](#) object.

4.25.3.3 fileNames()

```
int LibArchive::fileNames (
    const std::filesystem::path & filepath,
    std::vector< ArchEntry > & filenames)
```

Lists all entries in archive file.

Warning

This method is suitable for zip archives only. For other archive types behavior is undefined.

Parameters

<i>filepath</i>	absolute path to zip archive.
<i>filenames</i>	vector for results.

Returns

1 in case of success, -1 in case of error, 0 in case archive file has not been opened.

4.25.3.4 fileNamesStream()

```
int LibArchive::fileNamesStream (
    const std::filesystem::path & address,
    std::vector< ArchEntry > & filenames)
```

Lists all entries in archive file.

This method can be used with all supported archive types (see [AuxFunc::get_supported_archive_types_unpacking\(\)](#)). However for zip archives it is recommended to use [fileNames\(\)](#) method.

Parameters

<i>address</i>	absolute path to archive.
<i>filenames</i>	vector of results.

Returns

in case of succes returns ARCHIVE_OK, libarchive error codes will be returned otherwise (see archive.h file for details).

4.25.3.5 libarchive_error()

```
void LibArchive::libarchive_error (
    const std::shared_ptr< archive > & a,
    const std::string & message,
    const int & error_number)
```

Prints libarchive error messages.

Parameters

<i>a</i>	smart pointer to libarchive object.
<i>message</i>	extra text if needed (will be shown before libarchive error text).
<i>error_number</i>	error code.

4.25.3.6 libarchive_packing() [1/2]

```
int LibArchive::libarchive_packing (
    const std::filesystem::path & sourcepath,
    const std::filesystem::path & outpath)
```

Packs file or directory into archive.

Parameters

<i>sourcepath</i>	absolute path to file or directory to be packed.
<i>outpath</i>	absolute path to resulting archive.

Returns

-100 in case if sourcepath does not exist, -200 in case of error on libarchive object creation, ARCHIVE_OK in case of success, libarchive error code otherwise (see libarchive archive.h file for details).

4.25.3.7 libarchive_packing() [2/2]

```
int LibArchive::libarchive_packing (
    const std::shared_ptr< archive > & a,
    const std::filesystem::path & sourcepath,
    const bool & rename_source,
    const std::string & new_source_name)
```

Packs file or directory into archive.

Use this method if you need source file or directory to be packed under another name.

Parameters

<i>a</i>	smart pointer to libarchive object (see libarchive_write_init()).
<i>sourcepath</i>	absolute path to file or directory to be packed.
<i>rename_source</i>	if set to <i>true</i> , source name will be replaced for new_source_name inside the archive.
<i>new_source_name</i>	new name to be used inside the archive. Should be UTF-8 string.

Returns

-100 in case if sourcepath does not exist, -200 in case of error on libarchive object creation, ARCHIVE_OK in case of success, libarchive error code otherwise (see libarchive archive.h file for details).

4.25.3.8 libarchive_read_entry()

```
std::filesystem::path LibArchive::libarchive_read_entry (
    archive * a,
    archive_entry * entry,
    const std::filesystem::path & outfolder)
```

Unpacks libarchive entry content.

In most cases you do not need to use this method directly. Use [unpackByPosition\(\)](#), [unpackByPositionStr\(\)](#), [unpackByFileNameStream\(\)](#), [unpackByFileNameStreamStr\(\)](#) methods instead.

Parameters

<i>a</i>	pointer to libarchive object.
<i>entry</i>	pointer to libarchive entry object.
<i>outfolder</i>	directory entry content to be unpacked to. If this directory does not exist, it will be created.

Returns

Absolute path to unpacked file or directory.

4.25.3.9 libarchive_read_entry_str()

Parameters

<i>a</i>	pointer to libarchive object.
<i>entry</i>	entry to be read.

Returns

File content.

4.25.3.10 libarchive_read_init()

```
std::shared_ptr< archive > LibArchive::libarchive_read_init (
    std::shared_ptr< ArchiveFileEntry > fl)
```

Initializes archive reading.

This method can return *nullptr* in case of error. If this method failed, you can try to use [libarchive_read_init_fallback\(\)](#) instead.

Parameters

<i>fl</i>	smart pointer to ArchiveFileEntry object (see createArchFile()).
-----------	---

Returns

Smart pointer to libarchive object (can be *nullptr* in case of any error).

4.25.3.11 libarchive_read_init_fallback()

```
std::shared_ptr< archive > LibArchive::libarchive_read_init_fallback (
    std::shared_ptr< ArchiveFileEntry > fl)
```

Initializes archive reading.

This method can return *nullptr* in case of error.

Parameters

<i>fl</i>	smart pointer to ArchiveFileEntry object (see createArchFile()).
-----------	---

Returns

Smart pointer to libarchive object (can be *nullptr* in case of any error).

Parameters

<i>rm_e</i>	ArchiveRemoveEntry got from libarchive_remove_init() .
<i>to_remove</i>	list of entries to be removed.

Returns

ARCHIVE_OK in case of success, libarchive error code otherwise (see libarchive archive.h for details).

4.25.3.13 libarchive_remove_init()

```
ArchiveRemoveEntry LibArchive::libarchive_remove_init (
    const std::filesystem::path & sourcepath,
    const std::filesystem::path & outpath)
```

Initializes archive objects for removing entries from archive.

Parameters

<i>sourcepath</i>	absolute path to archive entries to be removed from.
<i>outpath</i>	absolute path to write new archive without removed entries.

Returns

[ArchiveRemoveEntry](#) object.

4.25.3.14 libarchive_write_data()

```
int LibArchive::libarchive_write_data (
    archive * a,
    const std::string & data)
```

Writes data to archive.

Writes raw data to archive. In most cases you do not need to call this method directly. Use [libarchive_write_directory\(\)](#) and [libarchive_write_file\(\)](#) methods instead.

Parameters

<i>a</i>	pointer to libarchive object.
<i>data</i>	data to be written.

Returns

libarchive error code (see libarchive archive.h for details).

4.25.3.15 libarchive_write_data_from_file()

```
int LibArchive::libarchive_write_data_from_file (  
    archive * a,  
    const std::filesystem::path & source)
```

Writes raw data from file to archive.

In most cases you do not need to use this method. Use [writeToArchive\(\)](#) instead.

Parameters

<i>a</i>	pointer to libarchive object.
<i>source</i>	absolute path to source file.

Returns

libarchive error code (see libarchive archive.h for details).

4.25.3.16 libarchive_write_directory()

```
int LibArchive::libarchive_write_directory (  
    archive * a,  
    archive_entry * entry,  
    const std::filesystem::path & path_in_arch,  
    const std::filesystem::path & source)
```

Writes directory to archive.

In most case you do not need to use this method. Use [writeToArchive\(\)](#) instead.

Note

This method resolves symbolic links and processes them as underlying objects.

Parameters

<i>a</i>	pointer to libarchive object.
<i>entry</i>	pointer to libarchive entry object.
<i>path_in_arch</i>	path of entry in archive (must be relative, example: "my_directory/my_file").
<i>source</i>	absolute path to directory to be packed.

Returns

ARCHIVE_OK in case of success, libarchive error code otherwise (see libarchive archive.h for details).

4.25.3.17 libarchive_write_file()

```
int LibArchive::libarchive_write_file (
    archive * a,
    archive_entry * entry,
    const std::filesystem::path & path_in_arch,
    const std::filesystem::path & source)
```

Writes file to archive.

In most case you do not need to use this method. Use [writeToArchive\(\)](#) instead.

Parameters

<i>a</i>	pointer to libarchive object.
<i>entry</i>	pointer to libarchive entry object.
<i>path_in_arch</i>	path of entry in archive (must be relative, example: "my_directory/my_file").
<i>source</i>	absolute path to directory to be packed.

Returns

ARCHIVE_OK in case of success, libarchive error code otherwise (see libarchive archive.h for details).

4.25.3.18 libarchive_write_init()

```
std::shared_ptr< archive > LibArchive::libarchive_write_init (
    const std::filesystem::path & outpath)
```

Initializes writing to archive.

Warning

If resulting archive path already exists, it will be overwritten.

Parameters

<i>outpath</i>	path to resulting archive.
----------------	----------------------------

Returns

Smart pointer to libarchive object (can be *nullptr* in case of any error).

4.25.3.19 unpackByFileNameStream()

Generated by Doxygen

```
std::filesystem::path LibArchive::unpackByFileNameStream (
    const std::filesystem::path & archaddress,
    const std::filesystem::path & outfolder.
```

Parameters

<i>archaddress</i>	absolute path to archive.
<i>outfolder</i>	absolute path to directory, entry to be unpacked to. If directory does not exist, it will be created.
<i>filename</i>	file or directory name in archive.

Returns

Absolute path to unpacked file or directory.

4.25.3.20 unpackByFileNameStreamStr()

```
std::string LibArchive::unpackByFileNameStreamStr (
    const std::filesystem::path & archaddress,
    const std::string & filename)
```

Unpacks entry content from archive.

If entry is a file, unpacks it and returns file content. If entry is a directory, returns empty string. This method is suitable for any supported types of archives (see [AuxFunc::get_supported_archive_types_unpacking\(\)](#)). However for zip archives it is recommended to use [unpackByPosition\(\)](#) and [unpackByPositionStr\(\)](#) methods (they are a little bit faster).

Parameters

<i>archaddress</i>	absolute path to archive.
<i>filename</i>	file or directory name in archive.

Returns

Unpacked file content or empty string.

4.25.3.21 unpackByPosition()

```
std::filesystem::path LibArchive::unpackByPosition (
    const std::filesystem::path & archaddress,
    const std::filesystem::path & outfolder,
    const ArchEntry & entry)
```

Unpacks single entry content from zip archive.

Access to entry is carried out by its absolute position in zip file. It is recommended to use this method for fast unpacking of single file or directory from zip archive.

Warning

This method should be used for zip archives only!

Parameters

<i>archaddress</i>	absolute path to zip archive.
<i>outfolder</i>	absolute path to directory archive entry content to be unpacked to. If directory does not exist, it will be created.
<i>entry</i>	ArchEntry object, obtained by fileNames() , fileNamesStream() or fileinfo() methods.

Returns

Absolute path to unpacked file or directory.

4.25.3.22 unpackByPositionStr()

```
std::string LibArchive::unpackByPositionStr (
    const std::filesystem::path & archaddress,
    const ArchEntry & entry)
```

Unpacks single entry content from zip archive.

If entry is a file, unpacks it and returns file content. If entry is a directory, returns empty string. Access to entry is carried out by its absolute position in zip file. It is recommended to use this method for fast unpacking of single file from zip archive.

Warning

This method should be used for zip archives only!

Parameters

<i>archaddress</i>	absolute path to zip archive.
<i>entry</i>	ArchEntry object, obtained by fileNames() , fileNamesStream() or fileinfo() methods.

Returns

Unpacked file content or empty string.

4.25.3.23 writeToArchive()

```
int LibArchive::writeToArchive (
    std::shared_ptr< archive > a,
    const std::filesystem::path & source,
    const std::filesystem::path & path_in_arch)
```

Writes file or directory to archive.

Generated by Doxygen

Note

This method resolves symbolic links and processes them as underlying objects.

Parameters

<i>a</i>	smart pointer to libarchive object (see libarchive_write_init()).
<i>source</i>	absolute path to file or directory to be packed.
<i>path_in_arch</i>	path of entry in archive (must be relative, example: my_directory/my_file).

Returns

ARCHIVE_OK in case of success, libarchive error code otherwise (see libarchive archive.h for details).

4.26 MLEException Class Reference

The [MLEException](#) class.

```
#include <MLEException.h>
```

Public Member Functions

- **MLEException ()**
MLEException constructor.
- **MLEException (const std::string &msg)**
MLEException constructor.
- **MLEException (const MLEException &other)**
MLEException copy constructor.
- **MLEException (MLEException &&other)**
MLEException move constructor.
- **MLEException & operator= (const MLEException &other)**
operator =
- **MLEException & operator= (MLEException &&other)**
operator =
- **operator bool ()**
Returns true if MLEException contains message.
- **std::string what ()**
Returns error message.

4.26.1 Detailed Description

The [MLEException](#) class.

This class is used as exception to indicate various errors.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 MLEException()

```
MLEException::MLEException (
    const std::string & msg)
```

[MLEException](#) constructor

Parameters

<i>msg</i>	message to be printed.
------------	------------------------

4.26.3 Member Function Documentation**4.26.3.1 what()**

```
std::string MLEException::what ()
```

Returns error message.

Returns

Error info message.

4.27 NotesBaseEntry Class Reference

The [NotesBaseEntry](#) class.

```
#include <NotesBaseEntry.h>
```

Public Member Functions

- **NotesBaseEntry** ()
NotesBaseEntry constructor.
- **NotesBaseEntry** (const std::string &collection_name, const std::filesystem::path &book_file_full_path, const std::string &book_path)
NotesBaseEntry constructor.
- **NotesBaseEntry** (const [NotesBaseEntry](#) &other)
NotesBaseEntry copy constructor.
- **NotesBaseEntry** ([NotesBaseEntry](#) &&other)
NotesBaseEntry move constructor.
- **NotesBaseEntry** & **operator=** (const [NotesBaseEntry](#) &other)
operator =
- **NotesBaseEntry** & **operator=** ([NotesBaseEntry](#) &&other)
operator =
- bool **operator==** (const [NotesBaseEntry](#) &other) const
operator ==

Public Attributes

- std::string **collection_name**
Collection name.
- std::filesystem::path **book_file_full_path**
Absolute path to book file.
- std::string **book_path**
Relative path to book in file (in case of archive, empty otherwise).
- std::filesystem::path **note_file_full_path**
Absolute path to note file.

4.27.1 Detailed Description

The [NotesBaseEntry](#) class.

Auxiliary class containing note info. In most cases you do not need to create [NotesBaseEntry](#) object yourself (see [NotesKeeper](#)).

4.27.2 Constructor & Destructor Documentation

4.27.2.1 NotesBaseEntry()

```
NotesBaseEntry::NotesBaseEntry (
    const std::string & collection_name,
    const std::filesystem::path & book_file_full_path,
    const std::string & book_path)
```

[NotesBaseEntry](#) constructor.

Parameters

<i>collection_name</i>	collection name.
<i>book_file_full_path</i>	absolute path to book file.
<i>book_path</i>	relative path to book in file (in case of archive, empty otherwise).

4.28 NotesKeeper Class Reference

The [NotesKeeper](#) class.

```
#include <NotesKeeper.h>
```

Public Member Functions

- [NotesKeeper](#) (const std::shared_ptr< [AuxFunc](#) > &af)
NotesKeeper constructor.
- virtual ~[NotesKeeper](#) ()
NotesKeeper destructor.
- void [loadBase](#) ()
Loads notes base to memory.
- void [editNote](#) (const [NotesBaseEntry](#) &nbe, const std::string ¬e)
Edits note.
- [NotesBaseEntry](#) [getNote](#) (const std::string &collection_name, const std::filesystem::path &book_file_full_path, const std::string &book_path)
Gets NotesBaseEntry object from base or creates it.
- std::string [readNote](#) (const [NotesBaseEntry](#) &nbe)
Returns content of note file.
- std::string [readNoteText](#) (const [NotesBaseEntry](#) &nbe)
Returns note text (if any).

- void [removeNotes](#) (const [NotesBaseEntry](#) &nbe, const std::filesystem::path &reserve_directory, const bool &make_reserve)
Removes notes.
- void [removeCollection](#) (const std::string &collection_name, const std::filesystem::path &reserve_directory, const bool &make_reserve)
Removes notes for all books in particular collection.
- void [refreshCollection](#) (const std::string &collection_name, const std::filesystem::path &reserve_directory, const bool &make_reserve)
Compares notes base and collection base and removes notes for absent books.
- std::vector< [NotesBaseEntry](#) > [getNotesForCollection](#) (const std::string &collection_name)
Returns all notes for particular collection.

4.28.1 Detailed Description

The [NotesKeeper](#) class.

This class contains various methods for collections notes operating. It is recommended to start from [loadBase\(\)](#) method. To create new note call [getNote\(\)](#) and [editNote\(\)](#) methods. [getNote\(\)](#) method also can be used to obtain note for particular book in collection. [editNote\(\)](#) method can be used to remove note.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 NotesKeeper()

```
NotesKeeper::NotesKeeper (
    const std::shared_ptr< AuxFunc > & af)
```

[NotesKeeper](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.28.3 Member Function Documentation

4.28.3.1 editNote()

```
void NotesKeeper::editNote (
    const NotesBaseEntry & nbe,
    const std::string & note)
```

Edits note.

If note file does not exist, this method will creat it and add [NotesBaseEntry](#) object to base. If note file exists, it will be overwritten. If note string is empty, note file will be removed, and [NotesBaseEntry](#) object will be removed from base.

Parameters

<i>nbe</i>	NotesBaseEntry object (see getNote() and getNotesForCollection()).
<i>note</i>	note text.

4.28.3.2 getNote()

```
NotesBaseEntry NotesKeeper::getNote (
    const std::string & collection_name,
    const std::filesystem::path & book_file_full_path,
    const std::string & book_path)
```

Gets [NotesBaseEntry](#) object from base or creates it.

If note exists, returns its [NotesBaseEntry](#) object. If note does not exist, creates new [NotesBaseEntry](#) object.

Parameters

<i>collection_name</i>	collection name, book came from.
<i>book_file_full_path</i>	book file absolute path.
<i>book_path</i>	book path in file (if any, empty string otherwise).

Returns

[NotesBaseEntry](#) object for note.

4.28.3.3 getNotesForCollection()

```
std::vector< NotesBaseEntry > NotesKeeper::getNotesForCollection (
    const std::string & collection_name)
```

Returns all notes for particular collection.

Parameters

<i>collection_name</i>	collection name.
------------------------	------------------

Returns

Vector of [NotesBaseEntry](#) objects.

4.28.3.4 loadBase()

```
void NotesKeeper::loadBase ()
```

Loads notes base to memory.

This method should be called before any other methods of this class.

4.28.3.5 readNote()

```
std::string NotesKeeper::readNote (  
    const NotesBaseEntry & nbe)
```

Returns content of note file.

Note file contains header and note text. Header contains collection name, book file absolute path, book path in file (if any). Header is separated from note text by "\n\n" sequence.

Parameters

<i>nbe</i>	NotesBaseEntry object (see getNote() and getNotesForCollection()).
------------	---

Returns

String containing note file raw content.

4.28.3.6 readNoteText()

```
std::string NotesKeeper::readNoteText (  
    const NotesBaseEntry & nbe)
```

Returns note text (if any).

Parameters

<i>nbe</i>	NotesBaseEntry object (see getNote() and getNotesForCollection()).
------------	---

Returns

String containing note text.

4.28.3.7 refreshCollection()

Generated by Doxygen

```
void NotesKeeper::refreshCollection (  
    const std::string & collection_name,  
    const std::filesystem::path & reserve_directory,  
    const bool & make_reserve)
```

Parameters

<i>collection_name</i>	collection to compare bases.
<i>reserve_directory</i>	absolute path to directory for reserve copies of notes to be removed. If directory does not exist, it will be created.
<i>make_reserve</i>	if set to <i>true</i> , refreshCollection() will create reserve copies of notes to be removed.

4.28.3.8 removeCollection()

```
void NotesKeeper::removeCollection (
    const std::string & collection_name,
    const std::filesystem::path & reserve_directory,
    const bool & make_reserve)
```

Removes notes for all books in particular collection.

Parameters

<i>collection_name</i>	collection name.
<i>reserve_directory</i>	absolute path to directory for reserve copies of notes to be removed. If directory does not exist, it will be created.
<i>make_reserve</i>	if set to <i>true</i> , removeCollection() will create reserve copies of notes to be removed.

4.28.3.9 removeNotes()

```
void NotesKeeper::removeNotes (
    const NotesBaseEntry & nbe,
    const std::filesystem::path & reserve_directory,
    const bool & make_reserve)
```

Removes notes.

It is recommended to use this method after book removing from collection.

Warning

If *nbe* parameter **book_file_full_path** contains path to rar archive, this method will remove notes for all books in archive.

Parameters

<i>nbe</i>	NotesBaseEntry object (see getNote() and getNotesForCollection()).
------------	---

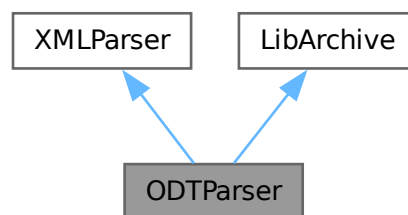
<code>reserve_directory</code>	absolute path to directory for reserve copies of notes to be removed. If directory does not exist, it will be created.
<code>make_reserve</code>	if set to <code>true</code> , <code>removeNotes()</code> will create reserve copies of notes to be removed.

4.29 ODTParser Class Reference

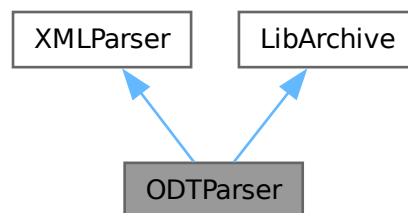
The `ODTParser` class.

```
#include <ODTParser.h>
```

Inheritance diagram for `ODTParser`:



Collaboration diagram for `ODTParser`:



Public Member Functions

- `ODTParser` (const std::shared_ptr< `AuxFunc` > &af)
ODTParser constructor.
- virtual `~ODTParser` ()
ODTParser destructor.
- `BookParseEntry odtParser` (const std::filesystem::path &odt_path)
Parses odt files.
- std::shared_ptr< `BookInfoEntry` > `odtBookInfo` (const std::filesystem::path &odt_path)
Gets some extra info from odt files.

Public Member Functions inherited from XMLParser

- [XMLParser](#) (const std::shared_ptr< [AuxFunc](#) > &af)
XMLParser constructor.
- std::vector< [XMLTag](#) > [get_tag](#) (const std::string &book, const std::string &tag_id)
Returns all tags with particular name.
- std::string [get_book_encoding](#) (const std::string &book)
Returns XML document encoding.
- std::string [get_element_attribute](#) (const std::string &element, const std::string &attr_name)
Returns XML tag attribute if it was found.
- std::vector< [XMLTag](#) > [listAllTags](#) (const std::string &book)
Parses XML document.
- void [searchTag](#) (const std::vector< [XMLTag](#) > &list, const std::string &tag_id, std::vector< [XMLTag](#) > &result)
Searches tag in tag list.
- void [htmlSymbolsReplacement](#) (std::string &book)
Replaces symbols encoded by "&..." sequences.
- void [removeAllTags](#) (std::string &book)
Removes all tag elements from XML document.

Public Member Functions inherited from LibArchive

- [LibArchive](#) (const std::shared_ptr< [AuxFunc](#) > &af)
LibArchive constructor.
- std::filesystem::path [unpackByPosition](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::string [unpackByPositionStr](#) (const std::filesystem::path &archaddress, const [ArchEntry](#) &entry)
Unpacks single entry content from zip archive.
- std::filesystem::path [unpackByFileNameStream](#) (const std::filesystem::path &archaddress, const std::filesystem::path &outfolder, const std::string &filename)
Unpacks entry content from archive.
- std::string [unpackByFileNameStreamStr](#) (const std::filesystem::path &archaddress, const std::string &filename)
Unpacks entry content from archive.
- int [fileNames](#) (const std::filesystem::path &filepath, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.
- int [fileNamesStream](#) (const std::filesystem::path &address, std::vector< [ArchEntry](#) > &filenames)
Lists all entries in archive file.
- [ArchEntry](#) [fileinfo](#) (const std::filesystem::path &address, const std::string &filename)
Returns ArchEntry for particular file or directory in archive.
- int [libarchive_packing](#) (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)
Packs file or directory into archive.
- int [libarchive_packing](#) (const std::shared_ptr< archive > &a, const std::filesystem::path &sourcepath, const bool &rename_source, const std::string &new_source_name)
Packs file or directory into archive.
- [ArchiveRemoveEntry](#) [libarchive_remove_init](#) (const std::filesystem::path &sourcepath, const std::filesystem::path &outpath)
Initializes archive objects for removing entries from archive.
- int [libarchive_remove_entry](#) ([ArchiveRemoveEntry](#) rm_e, const std::vector< [ArchEntry](#) > &to_remove)
Removes entry from archive.

- void [libarchive_error](#) (const std::shared_ptr< archive > &a, const std::string &message, const int &error_↵
number)
Prints libarchive error messages.
- std::string [libarchive_read_entry_str](#) (archive *a, archive_entry *entry)
Reads archived file to string.
- int [libarchive_write_data](#) (archive *a, const std::string &data)
Writes data to archive.
- std::shared_ptr< [ArchiveFileEntry](#) > [createArchFile](#) (const std::filesystem::path &archaddress, const la_↵
int64_t &position=la_int64_t(0))
Creates [ArchiveFileEntry](#) object.
- std::shared_ptr< archive > [libarchive_read_init](#) (std::shared_ptr< [ArchiveFileEntry](#) > fl)
Initializes archive reading.
- std::shared_ptr< archive > [libarchive_read_init_fallback](#) (std::shared_ptr< [ArchiveFileEntry](#) > fl)
Initializes archive reading.
- std::filesystem::path [libarchive_read_entry](#) (archive *a, archive_entry *entry, const std::filesystem::path &out-
folder)
Unpacks libarchive entry content.
- std::shared_ptr< archive > [libarchive_write_init](#) (const std::filesystem::path &outpath)
Initializes writing to archive.
- int [writeToArchive](#) (std::shared_ptr< archive > a, const std::filesystem::path &source, const std::filesystem↵
::path &path_in_arch)
Writes file or directory to archive.
- int [libarchive_write_directory](#) (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch,
const std::filesystem::path &source)
Writes directory to archive.
- int [libarchive_write_file](#) (archive *a, archive_entry *entry, const std::filesystem::path &path_in_arch, const
std::filesystem::path &source)
Writes file to archive.
- int [libarchive_write_data_from_file](#) (archive *a, const std::filesystem::path &source)
Writes raw data from file to archive.

4.29.1 Detailed Description

The [ODTParser](#) class.

This class contains methods for odt files processing. In most cases you do not need to use this class directly. Use [CreateCollection](#), [RefreshCollection](#) and [BookInfo](#) instead.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 ODTParser()

```
ODTParser::ODTParser (
    const std::shared_ptr< AuxFunc > & af)
```

[ODTParser](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.29.3 Member Function Documentation

4.29.3.1 odtBookInfo()

```
std::shared_ptr< BookInfoEntry > ODTParser::odtBookInfo (
    const std::filesystem::path & odt_path)
```

Gets some extra info from odt files.

This method can be used to obtain odt file cover and some other info (see [BookInfoEntry](#)) if such info is available.

Note

This method can throw [MLException](#) in case of some errors.

Parameters

<i>odt_path</i>	absolute path to odt file.
-----------------	----------------------------

Returns

Smart pointer to [BookInfoEntry](#) object.

4.29.3.2 odtParser()

```
BookParseEntry ODTParser::odtParser (
    const std::filesystem::path & odt_path)
```

Parses odt files.

This method can be used to obtain information from odt files.

Note

This method can throw [MLException](#) in case of some errors.

Parameters

<code>odt_path</code>	absolute path to odt file.
-----------------------	----------------------------

Returns

[BookParseEntry](#) object.

4.30 OmpLockGuard Class Reference

The [OmpLockGuard](#) class.

```
#include <OmpLockGuard.h>
```

Public Member Functions

- [OmpLockGuard](#) (omp_lock_t &omp_mtx)
OmpLockGuard constructor.
- virtual ~**OmpLockGuard** ()
OmpLockGuard destructor.

4.30.1 Detailed Description

The [OmpLockGuard](#) class.

Auxiliary class. Locks omp_lock_t variable on creation and unlocks it on destruction.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 OmpLockGuard()

```
OmpLockGuard::OmpLockGuard (
    omp_lock_t & omp_mtx)
```

[OmpLockGuard](#) constructor.

Warning

omp_mtx must be initialized (see [omp_init_lock](#)).

Parameters

<code>omp_mtx</code>	omp_lock_t variable.
----------------------	----------------------

4.31 OpenBook Class Reference

The [OpenBook](#) class.

```
#include <OpenBook.h>
```

Public Member Functions

- [OpenBook](#) (const std::shared_ptr< [AuxFunc](#) > &af)
OpenBook constructor.
- std::filesystem::path [open_book](#) (const [BookBaseEntry](#) &bbe, const bool ©, const std::filesystem::path ©_path, const bool &find_fbd, std::function< void(const std::filesystem::path &path)> open_callback)
Opens book.

4.31.1 Detailed Description

The [OpenBook](#) class.

This class contains methods for books "opening".

4.31.2 Constructor & Destructor Documentation

4.31.2.1 OpenBook()

```
OpenBook::OpenBook (
    const std::shared_ptr< AuxFunc > & af)
```

[OpenBook](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.31.3 Member Function Documentation

4.31.3.1 open_book()

```
std::filesystem::path OpenBook::open_book (
    const BookBaseEntry & bbe,
    const bool & copy,
    const std::filesystem::path & copy_path,
    const bool & find_fbd,
    std::function< void(const std::filesystem::path &path)> open_callback)
```

Opens book.

If book is in archive, unpacks book and returns absolute path to unpacked file. Otherwise returns absolute path to book file.

Parameters

<i>bbe</i>	BookBaseEntry object.
<i>copy</i>	if set to <i>true</i> , copy of book file will be created.
<i>copy_path</i>	absolute path to directory book to be copied to.
<i>find_fbd</i>	if set to <i>true</i> , this method will try to find and open fbd file instead of book.
<i>open_callback</i>	method to be called at the end of all operations. path argument is an absolute path to method work result.

Returns

Absolute path to book to be opened.

4.32 PaperBookInfoEntry Class Reference

The [PaperBookInfoEntry](#) class.

```
#include <PaperBookInfoEntry.h>
```

Public Member Functions

- **PaperBookInfoEntry ()**
PaperBookInfoEntry constructor.
- **PaperBookInfoEntry (const [PaperBookInfoEntry](#) &other)**
PaperBookInfoEntry copy constructor.
- **PaperBookInfoEntry ([PaperBookInfoEntry](#) &&other)**
PaperBookInfoEntry move constructor.
- **[PaperBookInfoEntry](#) & operator= (const [PaperBookInfoEntry](#) &other)**
operator =
- **[PaperBookInfoEntry](#) & operator= ([PaperBookInfoEntry](#) &&other)**
operator =

Public Attributes

- bool **available** = false
If paper book info is available, will be set to true.
- std::string **book_name**
Paper book name.
- std::string **publisher**
Paper book publisher.
- std::string **city**
City where paper book was published.
- std::string **year**
Year of paper book publishing.
- std::string **isbn**
ISBN of paper book.

4.32.1 Detailed Description

The [PaperBookInfoEntry](#) class.

Auxiliary class containing some information about paper book source (if any).

4.33 PDFParser Class Reference

The [PDFParser](#) class.

```
#include <PDFParser.h>
```

Public Member Functions

- [PDFParser](#) (const std::shared_ptr< [AuxFunc](#) > &af)
PDFParser constructor.
- [BookParseEntry pdf_parser](#) (const std::string &file)
Parses pdf file.
- std::shared_ptr< [BookInfoEntry](#) > [pdf_annotation_n_cover](#) (const std::string &file, const double &x_dpi, const double &y_dpi)
Returns pdf book annotation and cover.

4.33.1 Detailed Description

The [PDFParser](#) class.

This class contains methods for pdf book parsing, annotations and covers obtaining. In most cases you do not need to use this class directly. Use [CreateCollection](#), [RefreshCollection](#) and [BookInfo](#) instead.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 PDFParser()

```
PDFParser::PDFParser (
    const std::shared_ptr< AuxFunc > & af)
```

[PDFParser](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.33.3 Member Function Documentation

4.33.3.1 pdf_annotation_n_cover()

```
std::shared_ptr< BookInfoEntry > PDFParser::pdf_annotation_n_cover (
```

Parameters

<i>file</i>	pdf file content.
<i>x_dpi</i>	horizontal DPI .
<i>y_dpi</i>	vertical DPI .

Returns

Smart pointer to [BookInfoEntry](#) object.

4.33.3.2 pdf_parser()

```
BookParseEntry PDFParser::pdf_parser (
    const std::string & file)
```

Parses pdf file.

Parameters

<i>file</i>	pdf file content.
-------------	-------------------

Returns

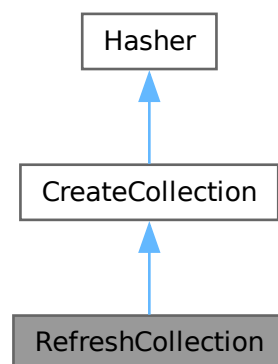
[BookParseEntry](#) object.

4.34 RefreshCollection Class Reference

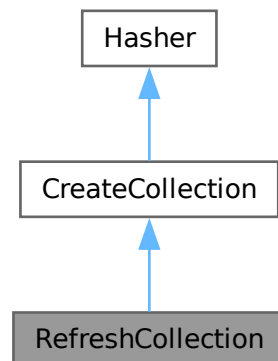
The [RefreshCollection](#) class.

```
#include <RefreshCollection.h>
```

Inheritance diagram for RefreshCollection:



Collaboration diagram for RefreshCollection:



Public Member Functions

- [RefreshCollection](#) (const std::shared_ptr< [AuxFunc](#) > &af, const std::string &collection_name, const int &num_threads, const bool &remove_empty, const bool &refresh_bookmarks, const bool &fast_refresh, const std::shared_ptr< [BookMarks](#) > &bookmarks)
[RefreshCollection](#) constructor.
- virtual ~**RefreshCollection** ()
[RefreshCollection](#) destructor.
- void [refreshCollection](#) ()
Refreshes whole collection.
- void [refreshFile](#) (const [BookBaseEntry](#) &bbe)
Refreshes iformation about particular file.
- bool [editBook](#) (const [BookBaseEntry](#) &bbe_old, const [BookBaseEntry](#) &bbe_new)
Replaces information in database.
- bool [refreshBook](#) (const [BookBaseEntry](#) &bbe)
Refreshes information in database about particular book.
- void [set_rar_support](#) (const bool &rar_support)
Enables support of rar archives.

Public Member Functions inherited from [CreateCollection](#)

- [CreateCollection](#) (const std::shared_ptr< [AuxFunc](#) > &af, const std::filesystem::path &collection_path, const std::filesystem::path &books_path, const bool &rar_support, const int &num_threads)
[CreateCollection](#) constructor.
- virtual ~**CreateCollection** ()
[CreateCollection](#) destructor.
- void [createCollection](#) ()
Starts collection creation.

Public Member Functions inherited from Hasher

- [Hasher](#) (const std::shared_ptr< [AuxFunc](#) > &af)
Hasher constructor.
- std::string [buf_hashing](#) (const std::string &buf)
Creates hash sum for given buffer.
- std::string [file_hashing](#) (const std::filesystem::path &filepath)
Creates hash sum for given file.
- void [cancelAll](#) ()
Stops all operations.

Public Attributes

- std::function< void(const double &total_hash)> [total_bytes_to_hash](#)
"Total bytes to hash" signal.
- std::function< void(const double &hashed)> [bytes_hashed](#)
"Total bytes hashed" signal.

Public Attributes inherited from CreateCollection

- std::function< void()> [pulse](#)
"Pulse" callback.
- std::function< void(const double &)> [signal_total_bytes](#)
"Total bytes" callback.
- std::function< void(const double &progress)> [progress](#)
"Progress" callback.

Additional Inherited Members

Protected Member Functions inherited from CreateCollection

- [CreateCollection](#) (const std::shared_ptr< [AuxFunc](#) > &af, const int &num_threads)
CreateCollection constructor.
- void [threadRegulator](#) ()
Threads regulator.
- void [openBaseFile](#) ()
Opens database file for writing.
- void [closeBaseFile](#) ()
Finishes database writing.
- void [write_file_to_base](#) (const [FileParseEntry](#) &fe)
Writes file data to database.

Protected Attributes inherited from [CreateCollection](#)

- `std::filesystem::path` [base_path](#)
Absolute path to database.
- `std::filesystem::path` [books_path](#)
Absolute path to books directory.
- `bool` [rar_support](#) = false
If true, rar archives will be processed, otherwise - not.
- `std::vector< std::tuple< std::filesystem::path, std::string > >` [already_hashed](#)
Hashed files.
- `std::vector< std::filesystem::path >` [need_to_parse](#)
"Need to parse" vector.
- `std::atomic< double >` [current_bytes](#)
Keeps quantity of bytes have been processed.

Protected Attributes inherited from [Hasher](#)

- `std::atomic< bool >` [cancel](#)
Stops all operations if true.
- `std::function< void()>` [stop_all_signal](#)
Stop signal for heir classes.

4.34.1 Detailed Description

The [RefreshCollection](#) class.

This class contains various methods for collection database refreshing in case of any changes were made to collection files.

4.34.2 Constructor & Destructor Documentation

4.34.2.1 [RefreshCollection\(\)](#)

```
RefreshCollection::RefreshCollection (
    const std::shared_ptr< AuxFunc > & af,
    const std::string & collection_name,
    const int & num_threads,
    const bool & remove_empty,
    const bool & refresh_bookmarks,
    const bool & fast_refresh,
    const std::shared_ptr< BookMarks > & bookmarks)
```

[RefreshCollection](#) constructor.

Note

See also [set_rar_support\(\)](#) method.

Parameters

<i>af</i>	smart pointer to AuxFunc object.
<i>collection_name</i>	collection name.
<i>num_threads</i>	number of threads to be used (see also CreateCollection::CreateCollection()).
<i>remove_empty</i>	if <i>true</i> , empty directories and files will be removed.
<i>fast_refresh</i>	if <i>true</i> , file hashes calculations will not be carried out (hashes of all collection files will be recalculated otherwise).
<i>refresh_bookmarks</i>	if <i>true</i> , bookmarks pointing to absent books will be removed.
<i>bookmarks</i>	smart pointer to BookMarks object.

4.34.3 Member Function Documentation**4.34.3.1 editBook()**

```
bool RefreshCollection::editBook (
    const BookBaseEntry & bbe_old,
    const BookBaseEntry & bbe_new)
```

Replaces information in database.

Use this method, if you need to edit database entries manually.

Parameters

<i>bbe_old</i>	existing BookBaseEntry (see BaseKeeper::searchBook()).
<i>bbe_new</i>	BookBaseEntry containing new information.

Returns

Returns *true*, if operation has been successful.

4.34.3.2 refreshBook()

```
bool RefreshCollection::refreshBook (
    const BookBaseEntry & bbe)
```

Refreshes information in database about particular book.

Parameters

<i>bbe</i>	BookBaseEntry object (see BaseKeeper::searchBook()).
------------	---

Returns

Returns *true*, if operation has been successful.

4.34.3.3 refreshCollection()

```
void RefreshCollection::refreshCollection ()
```

Refreshes whole collection.

Carries out collection refreshing.

Note

This method can throw [MLException](#) in case of errors.

4.34.3.4 refreshFile()

```
void RefreshCollection::refreshFile (
    const BookBaseEntry & bbe)
```

Refreshes information about particular file.

Parameters

<i>bbe</i>	BookBaseEntry object (see BaseKeeper::searchBook()).
------------	---

4.34.3.5 set_rar_support()

```
void RefreshCollection::set_rar_support (
    const bool & rar_support)
```

Enables support of rar archives.

Set ***rar_support*** to *true*, if you need to parse rar archives (see also [CreateCollection::CreateCollection\(\)](#)).

Parameters

<i>rar_support</i>	if <i>true</i> , rar archives will be parsed.
--------------------	---

4.34.4 Member Data Documentation

4.34.4.1 bytes_hashed

```
std::function<void(const double &hashed)> RefreshCollection::bytes_hashed
```

"Total bytes hashed" signal.

Emitted after file has been hashed, to indicate total quantity of bytes have been hashed. Bind your method to **bytes_hashed**, if you need such information.

4.34.4.2 total_bytes_to_hash

```
std::function<void(const double &total_hash)> RefreshCollection::total_bytes_to_hash
```

"Total bytes to hash" signal.

Emitted after files for refreshing have been collected, to indicate total quantity bytes to be hashed. Bind your method to **total_bytes_to_hash**, if you need such information.

4.35 RemoveBook Class Reference

The [RemoveBook](#) class.

```
#include <RemoveBook.h>
```

Public Member Functions

- [RemoveBook](#) (const std::shared_ptr< [AuxFunc](#) > &af, const [BookBaseEntry](#) &bbe, const std::string &col_name, const std::shared_ptr< [BookMarks](#) > &bookmarks)
RemoveBook constructor.
- void [removeBook](#) ()
Removes book.

4.35.1 Detailed Description

The [RemoveBook](#) class.

This class contains methods to carry out book removing from collection.

4.35.2 Constructor & Destructor Documentation

4.35.2.1 RemoveBook()

```
RemoveBook::RemoveBook (
    const std::shared_ptr< AuxFunc > & af,
    const BookBaseEntry & bbe,
    const std::string & col_name,
    const std::shared_ptr< BookMarks > & bookmarks)
```

[RemoveBook](#) constructor

Parameters

<i>af</i>	smart pointer to AuxFunc object.
<i>bbe</i>	BookBaseEntry containing book info.
<i>col_name</i>	collection name.
<i>bookmarks</i>	BookMarks object.

4.35.3 Member Function Documentation**4.35.3.1 removeBook()**

```
void RemoveBook::removeBook ()
```

Removes book.

Note

This method can throw [MLException](#) in case of errors.

4.36 ReplaceTagItem Class Reference

The [ReplaceTagItem](#) class.

```
#include <ReplaceTagItem.h>
```

Public Member Functions

- **ReplaceTagItem ()**
ReplaceTagItem constructor.
- **ReplaceTagItem** (const [ReplaceTagItem](#) &other)
ReplaceTagItem copy constructor.
- **ReplaceTagItem** ([ReplaceTagItem](#) &&other)
ReplaceTagItem move constructor.
- **ReplaceTagItem & operator=** (const [ReplaceTagItem](#) &other)
operator =
- **ReplaceTagItem & operator=** ([ReplaceTagItem](#) &&other)
operator =

Public Attributes

- std::string **tag_to_replace**
Id of tag to be replaced (see [XMLTag::tag_id](#)).
- std::string **begin_replacement**
Replacement for start tag element.
- std::string **end_replacement**
Replacement for end tag element.

4.36.1 Detailed Description

The [ReplaceTagItem](#) class.

Auxiliary class for [FormatAnnotation](#) (see [FormatAnnotation::setTagReplacementTable\(\)](#)).

4.37 SelfRemovingPath Class Reference

The [SelfRemovingPath](#) class.

```
#include <SelfRemovingPath.h>
```

Public Member Functions

- **SelfRemovingPath** ()
SelfRemovingPath constructor.
- virtual **~SelfRemovingPath** ()
SelfRemovingPath destructor.
- **SelfRemovingPath** (const [SelfRemovingPath](#) &other)
SelfRemovingPath copy constructor.
- **SelfRemovingPath** ([SelfRemovingPath](#) &&other)
SelfRemovingPath move constructor.
- [SelfRemovingPath](#) & **operator=** (const [SelfRemovingPath](#) &other)
operator =
- [SelfRemovingPath](#) & **operator=** ([SelfRemovingPath](#) &&other)
operator =
- [SelfRemovingPath](#) & **operator=** (const std::filesystem::path &path)
operator =
- **SelfRemovingPath** (const std::filesystem::path &path)
SelfRemovingPath constructor.

Public Attributes

- std::filesystem::path **path**
Path to be removed on destruction.

4.37.1 Detailed Description

The [SelfRemovingPath](#) class.

Auxiliary class. Removes underlying path on destruction, if no any copies of [SelfRemovingPath](#) object have been created. Removes path on last copy destruction otherwise.

4.37.2 Constructor & Destructor Documentation

4.37.2.1 SelfRemovingPath()

Generated by Doxygen

```
SelfRemovingPath::SelfRemovingPath (
    const std::filesystem::path & path) [explicit]
```

Parameters

<i>path</i>	path to be removed on destruction.
-------------	------------------------------------

4.37.3 Member Function Documentation**4.37.3.1 operator=()**

```
SelfRemovingPath & SelfRemovingPath::operator= (
    const std::filesystem::path & path)
```

operator =

Parameters

<i>path</i>	path to be removed on destruction.
-------------	------------------------------------

Returns

Returns self.

4.38 TXTParser Class Reference

The [TXTParser](#) class.

```
#include <TXTParser.h>
```

Public Member Functions

- [TXTParser](#) (const std::shared_ptr< [AuxFunc](#) > &af)
[TXTParser](#) constructor.
- [BookParseEntry txtParser](#) (const std::filesystem::path &txt_path)
Parses txt files.
- std::shared_ptr< [BookInfoEntry](#) > [txtBookInfo](#) (const std::filesystem::path &txt_path)
Gets some extra info from txt files.

4.38.1 Detailed Description

The [TXTParser](#) class.

This class contains methods for txt files processing. In most cases you do not need to use this class directly. Use [CreateCollection](#), [RefreshCollection](#) and [BookInfo](#) instead.

4.38.2 Constructor & Destructor Documentation

Parameters

<i>af</i>	smart pointer to AuxFunc object.
-----------	--

4.38.3 Member Function Documentation

4.38.3.1 txtBookInfo()

```
std::shared_ptr< BookInfoEntry > TXTParser::txtBookInfo (  
    const std::filesystem::path & txt_path)
```

Gets some extra info from txt files.

See also [BookInfoEntry](#).

Parameters

<i>txt_path</i>	absolute path to txt file.
-----------------	----------------------------

Returns

Smart pointer to [BookInfoEntry](#) object.

4.38.3.2 txtParser()

```
BookParseEntry TXTParser::txtParser (  
    const std::filesystem::path & txt_path)
```

Parses txt files.

This method can be used to obtain information from txt files.

Parameters

<i>txt_path</i>	absolute path to txt file.
-----------------	----------------------------

Returns

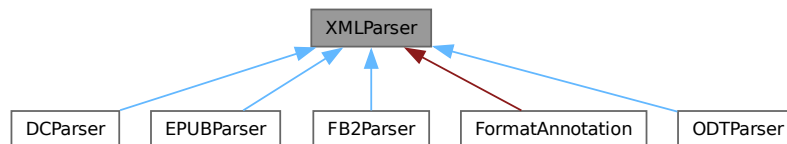
[BookParseEntry](#) object.

4.39 XMLParser Class Reference

The [XMLParser](#) class.

```
#include <XMLParser.h>
```

Inheritance diagram for XMLParser:



Public Member Functions

- [XMLParser](#) (const std::shared_ptr< [AuxFunc](#) > &af)
XMLParser constructor.
- std::vector< [XMLTag](#) > [get_tag](#) (const std::string &book, const std::string &tag_id)
Returns all tags with particular name.
- std::string [get_book_encoding](#) (const std::string &book)
Returns XML document encoding.
- std::string [get_element_attribute](#) (const std::string &element, const std::string &attr_name)
Returns XML tag attribute if it was found.
- std::vector< [XMLTag](#) > [listAllTags](#) (const std::string &book)
Parses XML document.
- void [searchTag](#) (const std::vector< [XMLTag](#) > &list, const std::string &tag_id, std::vector< [XMLTag](#) > &result)
Searches tag in tag list.
- void [htmlSymbolsReplacement](#) (std::string &book)
Replaces symbols encoded by "&..." sequences.
- void [removeAllTags](#) (std::string &book)
Removes all tag elements from XML document.

4.39.1 Detailed Description

The [XMLParser](#) class.

This class contains various methods to process XML documents.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 XMLParser()

```
XMLParser::XMLParser (
    const std::shared_ptr< AuxFunc > & af)
```

[XMLParser](#) constructor.

Parameters

<i>af</i>	smart pointer to AuxFunc constructor.
-----------	---

4.39.3 Member Function Documentation

4.39.3.1 `get_book_encoding()`

```
std::string XMLParser::get_book_encoding (  
    const std::string & book)
```

Returns [XML](#) document encoding.

Tries to get [XML](#) document header and read encoding from it.

Parameters

<i>book</i>	XML document content.
-------------	---------------------------------------

Returns

Returns [XML](#) document encoding if it was found.

4.39.3.2 `get_element_attribute()`

```
std::string XMLParser::get_element_attribute (  
    const std::string & element,  
    const std::string & attr_name)
```

Returns [XML](#) tag attribute if it was found.

Parameters

<i>element</i>	tag start element (see XMLTag::element).
<i>attr_name</i>	requested attribute name.

Returns

Returns attribute content if it was found.

4.39.3.3 `get_tag()`

```
std::vector< XMLTag > XMLParser::get_tag (  
    const std::string & book,
```

Parameters

<i>book</i>	XML document content.
<i>tag</i> ↔ <i>_id</i>	requested tag name.

Returns

Vector of found tags.

4.39.3.4 htmlSymbolsReplacement()

```
void XMLParser::htmlSymbolsReplacement (
    std::string & book)
```

Replaces symbols encoded by "&..." sequences.

Replaces symbols encoded by "&..." sequences for actual values (see [this](#) for details).

Parameters

<i>book</i>	XML document content symbols to be replaced in.
-------------	---

4.39.3.5 listAllTags()

```
std::vector< XMLTag > XMLParser::listAllTags (
    const std::string & book)
```

Parses [XML](#) document.

Returns all found tags.

Note

This method can throw [MLException](#) in case of errors.

Parameters

<i>book</i>	XML document content.
-------------	---------------------------------------

Returns

Vector of found tags.

Parameters

<i>book</i>	XML document content.
-------------	-----------------------

4.39.3.7 searchTag()

```
void XMLParser::searchTag (
    const std::vector< XMLTag > & list,
    const std::string & tag_id,
    std::vector< XMLTag > & result)
```

Searches tag in tag list.

Searches tags in tag list by given tag name.

Parameters

<i>list</i>	tag list search to be carried out in.
<i>tag↔ _id</i>	tag name for tags to be searched.
<i>result</i>	vector for results (can be not empty, this methods appends found tags to existing results).

4.40 XMLTag Class Reference

The XMLTag class.

```
#include <XMLTag.h>
```

Public Member Functions

- **XMLTag ()**
XMLTag constructor.
- **XMLTag (const XMLTag &other)**
XMLTag copy constructor.
- **XMLTag & operator= (const XMLTag &other)**
operator =
- **XMLTag (XMLTag &&other)**
XMLTag move constructor.
- **XMLTag & operator= (XMLTag &&other)**
operator =
- **bool hasContent () const**
Checks if tag has content.

Public Attributes

- `std::string` **element**
Tag start element content.
- `std::string` **tag_id**
Tag name.
- `std::string::size_type` **content_start**
Index of first byte of tag content.
- `std::string::size_type` **content_end**
Index of last byte of tag content.
- `std::vector< XMLTag >` **tag_list**
*List of **XML** tags, found in tag content (if any).*

4.40.1 Detailed Description

The `XMLTag` class.

Auxiliary class for `XMLParser`. Contains tag start element content, tag name, index of tag content first byte in **XML** document and index of content last byte. Also contains list of tags were found in tag content.

4.40.2 Member Function Documentation

4.40.2.1 hasContent()

```
bool XMLTag::hasContent () const
```

Checks if tag has content.

This method returns *true*, if **content_start** and **content_end** are not equal to `std::string::npos`.

Returns

true if tag has content.

4.40.3 Member Data Documentation

4.40.3.1 content_end

```
std::string::size_type XMLTag::content_end
```

Index of last byte of tag content.

Index of last byte of tag content in **XML** document. Can be equal to `std::string::npos`, if tag does not have content.

4.40.3.2 content_start

```
std::string::size_type XMLTag::content_start
```

Index of first byte of tag content.

Index of first byte of tag content in **XML** document. Start of tag can be found by **element** size subtraction from **content_start**. If **content_start** value is equal to `std::string::npos`, it indicates error on tag reading (even if tag does not have any content).

4.40.3.3 element

```
std::string XMLTag::element
```

Tag start element content.

Tag start element content including opening "<" and closing ">" symbols.

Index

- add_to_existing_archive
 - AddBook, [8](#)
- add_to_existing_archive_dir
 - AddBook, [8](#)
- AddBook, [7](#)
 - add_to_existing_archive, [8](#)
 - add_to_existing_archive_dir, [8](#)
 - AddBook, [8](#)
 - archive_filenames, [9](#)
 - overwrite_archive, [9](#)
 - overwrite_archive_dir, [9](#)
 - simple_add, [10](#)
 - simple_add_dir, [10](#)
- already_hashed
 - CreateCollection, [50](#)
- arch_parser
 - ARCHParser, [16](#)
- ArchEntry, [11](#)
- archive_filenames
 - AddBook, [9](#)
- ArchiveFileEntry, [11](#)
- ArchiveRemoveEntry, [12](#)
 - reset, [13](#)
- ARCHParser, [13](#)
 - arch_parser, [16](#)
 - ARCHParser, [15](#)
- auth_show_progr
 - BaseKeeper, [30](#)
- AuxFunc, [16](#)
 - copy_book_callback, [18](#)
 - create, [18](#)
 - detect_encoding, [18](#)
 - get_activated, [19](#)
 - get_charset_conv_quantity, [19](#)
 - get_converter_by_number, [19](#)
 - get_extension, [19](#)
 - get_genre_list, [20](#)
 - get_selfpath, [20](#)
 - get_supported_archive_types_packing, [20](#)
 - get_supported_archive_types_unpacking, [20](#)
 - get_supported_types, [20](#)
 - getDJVUContext, [21](#)
 - homePath, [21](#)
 - html_to_utf8, [21](#)
 - if_supported_type, [22](#)
 - ifSupportedArchivePackingType, [22](#)
 - ifSupportedArchiveUnpackaingType, [22](#)
 - libgcrypt_error_handling, [23](#)
 - open_book_callback, [23](#)
 - randomFileName, [23](#)
 - share_path, [24](#)
 - stringToLower, [24](#)
 - temp_path, [24](#)
 - time_t_to_date, [24](#)
 - to_hex, [25](#)
 - to_utf_8, [25](#)
 - utf8_to_system, [25](#)
 - utf_8_to, [26](#)
- base64
 - BookInfoEntry, [35](#)
- base_path
 - CreateCollection, [50](#)
- BaseKeeper, [26](#)
 - auth_show_progr, [30](#)
 - BaseKeeper, [27](#)
 - booksWithNotes, [27](#)
 - collectionAuthors, [28](#)
 - get_base_vector, [28](#)
 - get_books_path, [28](#)
 - getBooksQuantity, [29](#)
 - loadCollection, [29](#)
 - searchBook, [29](#)
- bgra
 - BookInfoEntry, [35](#)
- book_genre
 - BookParseEntry, [38](#)
- book_name
 - BookParseEntry, [38](#)
- book_path
 - BookParseEntry, [38](#)
- BookBaseEntry, [30](#)
 - BookBaseEntry, [31](#)
- BookInfo, [32](#)
 - BookInfo, [32](#)
 - get_book_info, [32](#)
 - set_dpi, [33](#)
- BookInfoEntry, [33](#)
 - base64, [35](#)
 - bgra, [35](#)
 - cover_types, [34](#)
 - error, [35](#)
 - file, [35](#)
 - rgb, [35](#)
 - rgba, [35](#)
 - text, [35](#)
- BookMarks, [35](#)
 - BookMarks, [35](#)
 - createBookMark, [36](#)

- getBookMarks, 36
 - removeBookMark, 36
- BookParseEntry, 37
 - book_genre, 38
 - book_name, 38
 - book_path, 38
- books
 - FileParseEntry, 67
- books_path
 - CreateCollection, 51
- booksWithNotes
 - BaseKeeper, 27
- buf_hashing
 - Hasher, 74
- ByteOrder, 38
 - ByteOrder, 40–42
 - get_big, 42
 - get_little, 42
 - get_native, 43
 - operator=, 43–45
 - set_big, 45
 - set_little, 45
- bytes_hashed
 - RefreshCollection, 109
- cancel
 - Hasher, 75
- closeBaseFile
 - CreateCollection, 49
- collectionAuthors
 - BaseKeeper, 28
- content_end
 - XMLTag, 118
- content_start
 - XMLTag, 118
- copy_book_callback
 - AuxFunc, 18
- cover_types
 - BookInfoEntry, 34
- create
 - AuxFunc, 18
- createArchFile
 - LibArchive, 78
- createBookMark
 - BookMarks, 36
- CreateCollection, 46
 - already_hashed, 50
 - base_path, 50
 - books_path, 51
 - closeBaseFile, 49
 - CreateCollection, 48, 49
 - createCollection, 49
 - current_bytes, 51
 - need_to_parse, 51
 - openBaseFile, 49
 - progress, 51
 - pulse, 51
 - rar_support, 52
 - signal_total_bytes, 52
 - threadRegulator, 50
 - write_file_to_base, 50
- createCollection
 - CreateCollection, 49
- current_bytes
 - CreateCollection, 51
- dcAuthor
 - DCParser, 54
- dcDate
 - DCParser, 55
- dcDescription
 - DCParser, 55
- dcGenre
 - DCParser, 55
- dcIdentifier
 - DCParser, 56
- dcLanguage
 - DCParser, 56
- DCParser, 52
 - dcAuthor, 54
 - dcDate, 55
 - dcDescription, 55
 - dcGenre, 55
 - dcIdentifier, 56
 - dcLanguage, 56
 - DCParser, 54
 - dcPublisher, 56
 - dcSource, 57
 - dcTitle, 57
- dcPublisher
 - DCParser, 56
- dcSource
 - DCParser, 57
- dcTitle
 - DCParser, 57
- detect_encoding
 - AuxFunc, 18
- djvu_book_info
 - DJVUParser, 58
- djvu_parser
 - DJVUParser, 59
- DJVUParser, 58
 - djvu_book_info, 58
 - djvu_parser, 59
 - DJVUParser, 58
- editBook
 - RefreshCollection, 107
- editNote
 - NotesKeeper, 91
- ElectroBookInfoEntry, 59
- element
 - XMLTag, 118
- epub_book_info
 - EPUBParser, 63
- epub_parser
 - EPUBParser, 63
- EPUBParser, 60

- epub_book_info, 63
- epub_parser, 63
- EPUBParser, 63
- error
 - BookInfoEntry, 35
- fb2_book_info
 - FB2Parser, 66
- fb2_parser
 - FB2Parser, 66
- FB2Parser, 64
 - fb2_book_info, 66
 - fb2_parser, 66
 - FB2Parser, 65
- file
 - BookInfoEntry, 35
- file_hash
 - FileParseEntry, 67
- file_hashing
 - Hasher, 75
- file_rel_path
 - FileParseEntry, 68
- fileinfo
 - LibArchive, 78
- fileNames
 - LibArchive, 79
- fileNamesStream
 - LibArchive, 79
- FileParseEntry, 67
 - books, 67
 - file_hash, 67
 - file_rel_path, 68
- final_cleaning
 - FormatAnnotation, 69
- FormatAnnotation, 68
 - final_cleaning, 69
 - FormatAnnotation, 69
 - remove_escape_sequences, 70
 - removeAllTags, 70
 - replace_tags, 70
 - setTagReplacementTable, 71
- Genre, 71
 - genre_code, 72
 - genre_name, 72
- genre_code
 - Genre, 72
- genre_name
 - Genre, 72
- GenreGroup, 72
 - group_name, 73
- get_activated
 - AuxFunc, 19
- get_base_vector
 - BaseKeeper, 28
- get_big
 - ByteOrder, 42
- get_book_encoding
 - XMLParser, 115
- get_book_info
 - BookInfo, 32
- get_books_path
 - BaseKeeper, 28
- get_charset_conv_quantity
 - AuxFunc, 19
- get_converter_by_number
 - AuxFunc, 19
- get_element_attribute
 - XMLParser, 115
- get_extension
 - AuxFunc, 19
- get_genre_list
 - AuxFunc, 20
- get_little
 - ByteOrder, 42
- get_native
 - ByteOrder, 43
- get_selfpath
 - AuxFunc, 20
- get_supported_archive_types_packing
 - AuxFunc, 20
- get_supported_archive_types_unpacking
 - AuxFunc, 20
- get_supported_types
 - AuxFunc, 20
- get_tag
 - XMLParser, 115
- getBookMarks
 - BookMarks, 36
- getBooksQuantity
 - BaseKeeper, 29
- getDJVUContext
 - AuxFunc, 21
- getNote
 - NotesKeeper, 92
- getNotesForCollection
 - NotesKeeper, 92
- group_name
 - GenreGroup, 73
- hasContent
 - XMLTag, 118
- Hasher, 73
 - buf_hashing, 74
 - cancel, 75
 - file_hashing, 75
 - Hasher, 74
 - stop_all_signal, 75
- homePath
 - AuxFunc, 21
- html_to_utf8
 - AuxFunc, 21
- htmlSymbolsReplacement
 - XMLParser, 116
- if_supported_type
 - AuxFunc, 22
- ifSupportedArchivePackingType

- AuxFunc, [22](#)
- ifSupportedArchiveUnpackaingType
 - AuxFunc, [22](#)
- LibArchive, [76](#)
 - createArchFile, [78](#)
 - fileinfo, [78](#)
 - fileNames, [79](#)
 - fileNamesStream, [79](#)
 - LibArchive, [78](#)
 - libarchive_error, [80](#)
 - libarchive_packing, [80](#)
 - libarchive_read_entry, [81](#)
 - libarchive_read_entry_str, [81](#)
 - libarchive_read_init, [82](#)
 - libarchive_read_init_fallback, [82](#)
 - libarchive_remove_entry, [82](#)
 - libarchive_remove_init, [83](#)
 - libarchive_write_data, [83](#)
 - libarchive_write_data_from_file, [83](#)
 - libarchive_write_directory, [84](#)
 - libarchive_write_file, [84](#)
 - libarchive_write_init, [85](#)
 - unpackByFileNameStream, [85](#)
 - unpackByFileNameStreamStr, [86](#)
 - unpackByPosition, [86](#)
 - unpackByPositionStr, [87](#)
 - writeToArchive, [87](#)
- libarchive_error
 - LibArchive, [80](#)
- libarchive_packing
 - LibArchive, [80](#)
- libarchive_read_entry
 - LibArchive, [81](#)
- libarchive_read_entry_str
 - LibArchive, [81](#)
- libarchive_read_init
 - LibArchive, [82](#)
- libarchive_read_init_fallback
 - LibArchive, [82](#)
- libarchive_remove_entry
 - LibArchive, [82](#)
- libarchive_remove_init
 - LibArchive, [83](#)
- libarchive_write_data
 - LibArchive, [83](#)
- libarchive_write_data_from_file
 - LibArchive, [83](#)
- libarchive_write_directory
 - LibArchive, [84](#)
- libarchive_write_file
 - LibArchive, [84](#)
- libarchive_write_init
 - LibArchive, [85](#)
- libgcrypt_error_handling
 - AuxFunc, [23](#)
- listAllTags
 - XMLParser, [116](#)
- loadBase
 - NotesKeeper, [92](#)
- loadCollection
 - BaseKeeper, [29](#)
- MLBookProc, [1](#)
- MLException, [88](#)
 - MLException, [88](#)
 - what, [89](#)
- need_to_parse
 - CreateCollection, [51](#)
- NotesBaseEntry, [89](#)
 - NotesBaseEntry, [90](#)
- NotesKeeper, [90](#)
 - editNote, [91](#)
 - getNote, [92](#)
 - getNotesForCollection, [92](#)
 - loadBase, [92](#)
 - NotesKeeper, [91](#)
 - readNote, [93](#)
 - readNoteText, [93](#)
 - refreshCollection, [93](#)
 - removeCollection, [94](#)
 - removeNotes, [94](#)
- odtBookInfo
 - ODTParser, [98](#)
- ODTParser, [95](#)
 - odtBookInfo, [98](#)
 - ODTParser, [97](#)
 - odtParser, [98](#)
- odtParser
 - ODTParser, [98](#)
- OmpLockGuard, [99](#)
 - OmpLockGuard, [99](#)
- open_book
 - OpenBook, [100](#)
- open_book_callback
 - AuxFunc, [23](#)
- openBaseFile
 - CreateCollection, [49](#)
- OpenBook, [100](#)
 - open_book, [100](#)
 - OpenBook, [100](#)
- operator=
 - ByteOrder, [43–45](#)
 - SelfRemovingPath, [112](#)
- overwrite_archive
 - AddBook, [9](#)
- overwrite_archive_dir
 - AddBook, [9](#)
- PaperBookInfoEntry, [101](#)
- pdf_annotation_n_cover
 - PDFParser, [102](#)
- pdf_parser
 - PDFParser, [103](#)
- PDFParser, [102](#)
 - pdf_annotation_n_cover, [102](#)

- pdf_parser, 103
- PDFParser, 102
- progress
 - CreateCollection, 51
- pulse
 - CreateCollection, 51
- randomFileName
 - AuxFunc, 23
- rar_support
 - CreateCollection, 52
- readNote
 - NotesKeeper, 93
- readNoteText
 - NotesKeeper, 93
- refreshBook
 - RefreshCollection, 107
- RefreshCollection, 103
 - bytes_hashed, 109
 - editBook, 107
 - refreshBook, 107
 - RefreshCollection, 106
 - refreshCollection, 108
 - refreshFile, 108
 - set_rar_support, 108
 - total_bytes_to_hash, 109
- refreshCollection
 - NotesKeeper, 93
 - RefreshCollection, 108
- refreshFile
 - RefreshCollection, 108
- remove_escape_sequences
 - FormatAnnotation, 70
- removeAllTags
 - FormatAnnotation, 70
 - XMLParser, 116
- RemoveBook, 109
 - RemoveBook, 109
 - removeBook, 110
- removeBook
 - RemoveBook, 110
- removeBookMark
 - BookMarks, 36
- removeCollection
 - NotesKeeper, 94
- removeNotes
 - NotesKeeper, 94
- replace_tags
 - FormatAnnotation, 70
- ReplaceTagItem, 110
- reset
 - ArchiveRemoveEntry, 13
- rgb
 - BookInfoEntry, 35
- rgba
 - BookInfoEntry, 35
- searchBook
 - BaseKeeper, 29
- searchTag
 - XMLParser, 117
- SelfRemovingPath, 111
 - operator=, 112
 - SelfRemovingPath, 111
- set_big
 - ByteOrder, 45
- set_dpi
 - BookInfo, 33
- set_little
 - ByteOrder, 45
- set_rar_support
 - RefreshCollection, 108
- setTagReplacementTable
 - FormatAnnotation, 71
- share_path
 - AuxFunc, 24
- signal_total_bytes
 - CreateCollection, 52
- simple_add
 - AddBook, 10
- simple_add_dir
 - AddBook, 10
- stop_all_signal
 - Hasher, 75
- stringToLower
 - AuxFunc, 24
- temp_path
 - AuxFunc, 24
- text
 - BookInfoEntry, 35
- threadRegulator
 - CreateCollection, 50
- time_t_to_date
 - AuxFunc, 24
- to_hex
 - AuxFunc, 25
- to_utf_8
 - AuxFunc, 25
- total_bytes_to_hash
 - RefreshCollection, 109
- txtBookInfo
 - TXTParser, 113
- TXTParser, 112
 - txtBookInfo, 113
 - TXTParser, 112
 - txtParser, 113
- txtParser
 - TXTParser, 113
- unpackByFileNameStream
 - LibArchive, 85
- unpackByFileNameStreamStr
 - LibArchive, 86
- unpackByPosition
 - LibArchive, 86
- unpackByPositionStr
 - LibArchive, 87

- utf8_to_system
 - AuxFunc, [25](#)
- utf_8_to
 - AuxFunc, [26](#)
- what
 - MLEException, [89](#)
- write_file_to_base
 - CreateCollection, [50](#)
- writeToArchive
 - LibArchive, [87](#)
- XMLParser, [114](#)
 - get_book_encoding, [115](#)
 - get_element_attribute, [115](#)
 - get_tag, [115](#)
 - htmlSymbolsReplacement, [116](#)
 - listAllTags, [116](#)
 - removeAllTags, [116](#)
 - searchTag, [117](#)
 - XMLParser, [114](#)
- XMLTag, [117](#)
 - content_end, [118](#)
 - content_start, [118](#)
 - element, [118](#)
 - hasContent, [118](#)