

Irc - Linux Resource Compiler
1.0

Generated by Doxygen 1.8.1.2

Sun Jan 13 2013 14:48:53

Contents

1	Irc - Linux Resource Compiler	1
1.1	Introduction	1
1.2	Compiler Options	1
1.3	The library liblrc	1
1.4	Example	1
2	Todo List	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	11
6.1	File List	11
7	Namespace Documentation	13
7.1	Irc Namespace Reference	13
7.1.1	Detailed Description	13
7.1.2	Enumeration Type Documentation	13
7.1.2.1	CompressionType	13
7.1.2.2	EncryptionType	14
8	Class Documentation	15
8.1	bz2LibCompression Class Reference	15
8.1.1	Detailed Description	15
8.1.2	Member Function Documentation	15
8.1.2.1	compress	15
8.1.2.2	decompress	16
8.2	Collector Class Reference	16

8.2.1	Detailed Description	16
8.2.2	Constructor & Destructor Documentation	17
8.2.2.1	Collector	17
8.2.2.2	~Collector	17
8.2.3	Member Function Documentation	17
8.2.3.1	are_resIDs_unique	17
8.2.3.2	collect	18
8.2.3.3	show_resource_data_error	18
8.2.4	Member Data Documentation	18
8.2.4.1	m_rcName	18
8.2.4.2	m_rdfName	19
8.3	Irc::CompressDecompress Class Reference	19
8.3.1	Detailed Description	19
8.3.2	Member Function Documentation	19
8.3.2.1	compress	19
8.3.2.2	decompress	20
8.4	CompressionFactory Class Reference	20
8.4.1	Detailed Description	20
8.4.2	Member Function Documentation	21
8.4.2.1	get_compression_class	21
8.5	Irc::EncryptDecrypt Class Reference	21
8.5.1	Detailed Description	21
8.5.2	Member Function Documentation	21
8.5.2.1	decrypt	21
8.5.2.2	encrypt	22
8.6	EncryptionFactory Class Reference	22
8.6.1	Detailed Description	23
8.6.2	Member Function Documentation	23
8.6.2.1	get_encryption_class	23
8.7	InFileParser Class Reference	23
8.7.1	Detailed Description	24
8.7.2	Member Enumeration Documentation	25
8.7.2.1	internalErrorType	25
8.7.3	Constructor & Destructor Documentation	25
8.7.3.1	InFileParser	25
8.7.3.2	~InFileParser	25
8.7.4	Member Function Documentation	25
8.7.4.1	clear_internal_error	25
8.7.4.2	eval_compression_type	26
8.7.4.3	eval_encryption_type	26

8.7.4.4	get_internal_error	26
8.7.4.5	get_password	27
8.7.4.6	get_resource_entries	27
8.7.4.7	parse	27
8.7.5	Member Data Documentation	28
8.7.5.1	m_errorPosition	28
8.7.5.2	m_filename	28
8.7.5.3	m_internalError	28
8.7.5.4	m_lastError	28
8.7.5.5	m_resEntries	28
8.8	IrcEncryptionDisabledException Class Reference	28
8.8.1	Detailed Description	29
8.8.2	Constructor & Destructor Documentation	29
8.8.2.1	IrcEncryptionDisabledException	29
8.8.2.2	~IrcEncryptionDisabledException	29
8.8.3	Member Function Documentation	29
8.8.3.1	what	29
8.8.4	Member Data Documentation	29
8.8.4.1	m_resourceID	29
8.9	IrcFileExistsException Class Reference	30
8.9.1	Detailed Description	30
8.9.2	Constructor & Destructor Documentation	30
8.9.2.1	IrcFileExistsException	30
8.9.2.2	~IrcFileExistsException	30
8.9.3	Member Function Documentation	30
8.9.3.1	what	30
8.9.4	Member Data Documentation	31
8.9.4.1	m_fileOverwrite	31
8.10	IrcFileNotFoundException Class Reference	31
8.10.1	Detailed Description	31
8.10.2	Constructor & Destructor Documentation	31
8.10.2.1	IrcFileNotFoundException	31
8.10.2.2	~IrcFileNotFoundException	32
8.10.3	Member Function Documentation	32
8.10.3.1	what	32
8.10.4	Member Data Documentation	32
8.10.4.1	m_fileNotFound	32
8.11	NoneCompression Class Reference	32
8.11.1	Detailed Description	32
8.11.2	Member Function Documentation	33

8.11.2.1	compress	33
8.11.2.2	decompress	33
8.12	NoneEncryption Class Reference	33
8.12.1	Detailed Description	33
8.12.2	Member Function Documentation	33
8.12.2.1	decrypt	34
8.12.2.2	encrypt	34
8.13	ParserFactory Class Reference	34
8.13.1	Detailed Description	34
8.13.2	Member Function Documentation	34
8.13.2.1	create_input_parser	34
8.14	RCParse Class Reference	35
8.14.1	Detailed Description	35
8.14.2	Constructor & Destructor Documentation	35
8.14.2.1	RCParse	35
8.14.3	Member Function Documentation	36
8.14.3.1	is_comment	36
8.14.3.2	parse	36
8.15	resEntry_ Struct Reference	36
8.15.1	Detailed Description	37
8.15.2	Member Data Documentation	37
8.15.2.1	compType	37
8.15.2.2	encType	37
8.15.2.3	resID	37
8.15.2.4	resSize	37
8.15.2.5	startOffset	37
8.16	Irc::Resource Class Reference	38
8.16.1	Detailed Description	38
8.16.2	Constructor & Destructor Documentation	38
8.16.2.1	Resource	38
8.16.2.2	~Resource	39
8.16.3	Member Function Documentation	39
8.16.3.1	get_ID	39
8.16.3.2	get_res_data	39
8.16.3.3	get_res_size	39
8.16.4	Member Data Documentation	39
8.16.4.1	m_resData	39
8.16.4.2	m_resID	39
8.16.4.3	m_resSize	40
8.17	ResourceData Class Reference	40

8.17.1	Detailed Description	41
8.17.2	Constructor & Destructor Documentation	41
8.17.2.1	ResourceData	41
8.17.2.2	~ResourceData	41
8.17.3	Member Function Documentation	41
8.17.3.1	get_compression	41
8.17.3.2	get_data_from_memory	42
8.17.3.3	get_encryption	42
8.17.3.4	get_error_msg	42
8.17.3.5	get_file	43
8.17.3.6	get_rc_position	43
8.17.3.7	prepare_resource_from_file	43
8.17.3.8	set_compression	44
8.17.3.9	set_encryption	44
8.17.3.10	set_error_msg	44
8.17.3.11	set_file	44
8.17.3.12	set_ident	44
8.17.3.13	set_rc_position	44
8.17.4	Member Data Documentation	45
8.17.4.1	m_compression	45
8.17.4.2	m_encryption	45
8.17.4.3	m_errorMsg	45
8.17.4.4	m_filename	45
8.17.4.5	m_inFilePosition	45
8.17.4.6	m_password	45
8.18	Irc::ResourceManager Class Reference	46
8.18.1	Detailed Description	46
8.18.2	Constructor & Destructor Documentation	47
8.18.2.1	ResourceManager	47
8.18.2.2	~ResourceManager	47
8.18.3	Member Function Documentation	47
8.18.3.1	get_resource	47
8.18.3.2	get_resource	47
8.18.3.3	get_resource_ids	48
8.18.3.4	load_from_file	48
8.18.4	Member Data Documentation	48
8.18.4.1	m_compType	48
8.18.4.2	m_encType	48
8.18.4.3	m_numResEntries	48
8.18.4.4	m_password	49

8.18.4.5	m_resData	49
8.18.4.6	m_resDataSize	49
8.18.4.7	m_resEntries	49
8.18.4.8	m_resourceFile	49
8.19	RIFParser Class Reference	49
8.19.1	Detailed Description	50
8.19.2	Constructor & Destructor Documentation	50
8.19.2.1	RIFParser	50
8.19.3	Member Function Documentation	50
8.19.3.1	parse	50
8.20	SerpentEncryption Class Reference	50
8.20.1	Detailed Description	51
8.20.2	Member Function Documentation	51
8.20.2.1	create_initialization_vector	51
8.20.2.2	decrypt	51
8.20.2.3	encrypt	52
8.21	zLibCompression Class Reference	52
8.21.1	Detailed Description	52
8.21.2	Member Function Documentation	52
8.21.2.1	compress	52
8.21.2.2	decompress	52
9	File Documentation	55
9.1	/home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx File Reference	55
9.1.1	Detailed Description	55
9.2	/home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx File Reference	55
9.2.1	Detailed Description	56
9.3	/home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx File Reference	56
9.3.1	Detailed Description	56
9.3.2	Macro Definition Documentation	57
9.3.2.1	VERSION	57
9.3.3	Function Documentation	57
9.3.3.1	main	57
9.3.3.2	usage	57
9.4	/home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx File Reference	57
9.4.1	Detailed Description	57
9.5	/home/andy/Programming/Projects/lrc/src/compiler/RCParse.hxx File Reference	57
9.5.1	Detailed Description	58
9.6	/home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx File Reference	58
9.6.1	Detailed Description	58

9.7	/home/andy/Programming/Projects/lrc/src/Factories.hxx File Reference	58
9.7.1	Detailed Description	59
9.8	/home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx File Reference	59
9.8.1	Detailed Description	59
9.9	/home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx File Reference	60
9.9.1	Detailed Description	60
9.10	/home/andy/Programming/Projects/lrc/src/include/Resource.hxx File Reference	60
9.10.1	Detailed Description	61
9.10.2	Macro Definition Documentation	61
9.10.2.1	MAX_ID_LEN	61
9.10.3	Typedef Documentation	61
9.10.3.1	resEntry	61
9.11	/home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx File Reference	61
9.11.1	Detailed Description	62
9.12	/home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx File Reference	62
9.12.1	Detailed Description	62
9.13	/home/andy/Programming/Projects/lrc/src/ResourceData.hxx File Reference	63
9.13.1	Detailed Description	63
9.13.2	Typedef Documentation	63
9.13.2.1	inFilePosition	63
9.14	/home/andy/Programming/Projects/lrc/src/StatusCodes.hxx File Reference	63
9.14.1	Detailed Description	64
9.14.2	Macro Definition Documentation	65
9.14.2.1	ERROR_BASE	65
9.14.2.2	ERROR_COMPRESSION_COMPRESS	65
9.14.2.3	ERROR_COMPRESSION_DECOMPRESS	65
9.14.2.4	ERROR_COMPRESSION_NOT_AVAILABLE	65
9.14.2.5	ERROR_ENCRYPTION_DECRYPT	65
9.14.2.6	ERROR_ENCRYPTION_ENCRYPT	65
9.14.2.7	ERROR_ENCRYPTION_NOT_AVAILABLE	65
9.14.2.8	ERROR_FILE_NOT_FOUND	65
9.14.2.9	ERROR_FILE_OPEN	65
9.14.2.10	ERROR_FILE_READ	66
9.14.2.11	ERROR_FILE_WRITE	66
9.14.2.12	ERROR_INVALID_PARAMETER	66
9.14.2.13	ERROR_NOT_ENOUGH_MEMORY	66
9.14.2.14	ERROR_PARSE	66
9.14.2.15	NO_ERROR	66
9.14.2.16	WARNING_BASE	66
9.14.2.17	WARNING_DOUBLE_RESOURCE_ID	66

9.14.3	Function Documentation	66
9.14.3.1	is_error	66
9.14.3.2	is_warning	67
9.14.3.3	no_error	67
9.14.3.4	success	67
9.15	/home/andy/Programming/Projects/lrc/src/strategies/bz2LibCompression.hxx File Reference	68
9.15.1	Detailed Description	68
9.16	/home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.hxx File Reference	68
9.16.1	Detailed Description	68
9.17	/home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx File Reference	69
9.17.1	Detailed Description	69
9.18	/home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx File Reference	69
9.18.1	Detailed Description	70
9.19	/home/andy/Programming/Projects/lrc/src/strategies/zLibCompression.hxx File Reference	70
9.19.1	Detailed Description	70
9.20	/home/andy/Programming/Projects/lrc/src/Utils.hxx File Reference	70
9.20.1	Detailed Description	71
9.20.2	Macro Definition Documentation	71
9.20.2.1	DEBUG_PRINT	71
9.20.3	Function Documentation	71
9.20.3.1	delete_list	71
9.20.3.2	file_exists	72
9.20.3.3	file_size	72
9.20.3.4	get_extension	72
9.20.3.5	password_len	72
9.20.3.6	replace_extension	73

Chapter 1

Irc - Linux Resource Compiler

1.1 Introduction

This documentation covers the classes and data structures of the `lrc` compiler and the library `liblrc`.

1.2 Compiler Options

The compiler needs at least one input file. It is called as follows:

```
lrc -h | [-d] [-o <rdfFile>] [-c <compression type>]<RCFile>|<rifFile>
```

The options have the following meaning:

`-h`: Show the usage of the compiler and quit.

`-d`: Deny overwrite an existing `.rdf` file. This parameter is optional and the default is overwriting allowed. If `-d` is set and the file already exists, the compiler exits with an error message.

`-o <file>`: Specify the resource output file (`.rdf`). This parameter is optional. If it is omitted the output file will be the name of the input file, but with `.rdf` as file extension.

`-c <compression type>`: Specify the compression type for the whole `.rdf` file. Allowed are all compression types that are allowed for a single resource file. These are at the moment:

- `zLib`: `zLib` compression
- `bzip2`: `bzip2` compression

1.3 The library `liblrc`

The library consists mainly of two classes: the [ResourceManager](#) and the [Resource](#) class. One `ResourceManager` instance is used for one resource data file (`.rdf`).

The resource manager provides two main methods: one lists all resource entries from the `.rdf` file, the other returns an instance of a `Resource` class, identified by its ID or index.

1.4 Example

A few examples that demonstrates the `lrc` and `liblrc` can be found in the `src/example` directory.

Author

Andreas Tschärner

Date

2013-01-13

Chapter 2

Todo List

Member [Irc::CompressionType](#)

Add more possibilities to compress resource

Member [Irc::EncryptionType](#)

Add more possibilities to encrypt resource

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

lrc	Namespace lrc for classes in the library and for extending lrc	13
---------------------	--	--------------------

Chapter 4

Class Index

4.1 Class Hierarchy

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

Collector	16
Irc::CompressDecompress	19
bz2LibCompression	15
NoneCompression	32
zLibCompression	52
CompressionFactory	20
Irc::EncryptDecrypt	21
NoneEncryption	33
SerpentEncryption	50
EncryptionFactory	22
InFileParser	23
RCParser	35
RIFParser	49
IrcEncryptionDisabledException	28
IrcFileExistsException	30
IrcFileNotFoundException	31
ParserFactory	34
resEntry_	36
Irc::Resource	38
ResourceData	40
Irc::ResourceManager	46

Chapter 5

Class Index

5.1 Class List

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

bz2LibCompression	Compression class that uses the <i>bzip2</i> algorithm	15
Collector	Class to collect all resource data	16
Irc::CompressDecompress	Compression and Decompression base class	19
CompressionFactory	Factory to create compression class instances	20
Irc::EncryptDecrypt	Encryption and Decryption base class	21
EncryptionFactory	Factory to create encryption class instances	22
InFileParser	Base class for all <code>lrc</code> input files	23
IrcEncryptionDisabledException	Exception if encryption is disabled but required	28
IrcFileExistsException	Exception if an existing file should be overwritten	30
IrcFileNotFoundException	Exception class if a file could not be found	31
NoneCompression	Compression class that does <i>NO</i> compression	32
NoneEncryption	Encryption class that does <i>NO</i> encryption	33
ParserFactory	Factory class to create an appropriate parser class	34
RCParser	Class to parse an <code>.rc</code> file	35
resEntry_	Data for one resource entry	36
Irc::Resource	Resource class for use with library	38
ResourceData	Internal class for resource with more information	40
Irc::ResourceManager	Manager class handling all resources of a resource file	46
RIFParser	Class to parse a <code>.rif</code> (XML) file	49

SerpentEncryption	
Class to encrypt/decrypt using the <i>Serpent</i> algorithm	50
zLibCompression	
Compression class that uses the <i>zLib</i> algorithm	52

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

/home/andy/Programming/Projects/lrc/src/Factories.hxx	58
/home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx	62
/home/andy/Programming/Projects/lrc/src/ResourceData.hxx	63
/home/andy/Programming/Projects/lrc/src/StatusCodes.hxx	63
/home/andy/Programming/Projects/lrc/src/Utils.hxx	70
/home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx	55
/home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx	55
/home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx	56
/home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx	57
/home/andy/Programming/Projects/lrc/src/compiler/RCParseR.hxx	57
/home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx	58
/home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx	59
/home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx	60
/home/andy/Programming/Projects/lrc/src/include/Resource.hxx	60
/home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx	61
/home/andy/Programming/Projects/lrc/src/strategies/bz2LibCompression.hxx	68
/home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.hxx	68
/home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx	69
/home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx	69
/home/andy/Programming/Projects/lrc/src/strategies/zLibCompression.hxx	70

Chapter 7

Namespace Documentation

7.1 Irc Namespace Reference

Namespace Irc for classes in the library and for extending `irc`.

Classes

- class [CompressDecompress](#)
Compression and Decompression base class.
- class [EncryptDecrypt](#)
Encryption and Decryption base class.
- class [Resource](#)
Resource class for use with library.
- class [ResourceManager](#)
Manager class handling all resources of a resource file.

Enumerations

- enum [CompressionType](#) { [NoneCompression](#), [zLibCompression](#), [bz2LibCompression](#) }
- enum [EncryptionType](#) { [NoneEncryption](#), [SerpentEncryption](#) }

7.1.1 Detailed Description

Namespace `irc` for classes in the library and for extending `irc`. The namespace `irc` was introduced and used for classes and functions that will be used in the `lib1irc` library to avoid confusion with other classes that may have the same name.

It is also used for two abstract classes: [CompressDecompress](#) and [EncryptDecrypt](#). These two classes are the base classes for compression and decompression and for encryption and decryption of the resource data. They can be found in the [CompressDecompress.hxx](#) and the [EncryptDecrypt.hxx](#) file respectively.

7.1.2 Enumeration Type Documentation

7.1.2.1 enum `irc::CompressionType`

Possible compression types for resource

```
\xrefitem todo 1.
```

Enumerator:

NoneCompression No compression at all.

zLibCompression zlib compression

bz2LibCompression bzip2 compression

Definition at line 44 of file CompressDecompress.hxx.

7.1.2.2 enum Irc::EncryptionType

Possible encryption types for resource

```
\xrefitem todo 2.
```

Enumerator:

NoneEncryption No encryption at all.

SerpentEncryption Serpent algorithm for encryption.

Definition at line 44 of file EncryptDecrypt.hxx.

Chapter 8

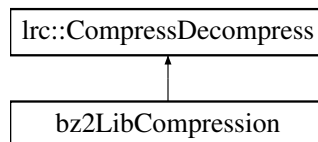
Class Documentation

8.1 bz2LibCompression Class Reference

Compression class that uses the *bzip2* algorithm.

```
#include <bz2LibCompression.hxx>
```

Inheritance diagram for bz2LibCompression:



Public Member Functions

- int [compress](#) (const unsigned char *, size_t, unsigned char **, size_t &)
bzip2 compression
- int [decompress](#) (const unsigned char *, size_t, unsigned char **, size_t &)
bzip2 decompression

8.1.1 Detailed Description

Compression class that uses the *bzip2* algorithm.

This class uses the bzip2 algorithm for compression and decompression of the resource data.

Definition at line 45 of file bz2LibCompression.hxx.

8.1.2 Member Function Documentation

8.1.2.1 int bz2LibCompression::compress (const unsigned char *, size_t, unsigned char **, size_t &) [virtual]

bzip2 compression

This method compresses the given data using the bzip2 algorithm

Implements [Irc::CompressDecompress](#).

8.1.2.2 `int bz2LibCompression::decompress (const unsigned char *, size_t, unsigned char **, size_t &) [virtual]`

bzip2 decompression

This method decompresses the given data using the bzip2 algorithm

Implements [Irc::CompressDecompress](#).

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

- [/home/andy/Programming/Projects/Irc/src/strategies/bz2LibCompression.hxx](#)

8.2 Collector Class Reference

Class to collect all resource data.

```
#include <Collector.hxx>
```

Public Member Functions

- [Collector](#) (char *, char *, bool) throw (IrcFileExistsException)
Constructor.
- [~Collector](#) (void)
Destructor.
- int [collect](#) (std::vector< [ResourceData](#) * > *, [Irc::CompressionType](#), [Irc::EncryptionType](#), const unsigned char *)
Collect, compress and encrypt data.

Private Member Functions

- int [are_resIDs_unique](#) (std::vector< [ResourceData](#) * >, [inFilePosition](#) &, char **)
Check for unique resource IDs.
- void [show_resource_data_error](#) (int, [ResourceData](#) *)
Show error message from resource data.

Private Attributes

- char * [m_rcName](#)
Filename of .rc file.
- char * [m_rdfName](#)
Filename for .rdf file.

8.2.1 Detailed Description

Class to collect all resource data.

This class collects all files that are defined as resource data and generates one big file from them

Definition at line 47 of file [Collector.hxx](#).

8.2.2 Constructor & Destructor Documentation

8.2.2.1 Collector::Collector (char *, char *, bool) throw (IrcFileExistsException)

Constructor.

This constructor is responsible to set up the class for collecting the resource data. It expects a filename for the .rdf file and a flag indicating whether or not overwriting is allowed

Parameters

in	<i>p_rcName</i>	Filename of rc file
in	<i>p_rdfName</i>	Filename for .rdf file
in	<i>p_overwriteAllow</i>	Flag indicating that overwriting is allowed (or not)

Exceptions

<i>IrcFileExistsException</i>	Exception that is thrown if the given .rdf file exists and the overwrite flag indicates that overwriting is forbidden
---	---

8.2.2.2 Collector::~~Collector (void)

Destructor.

Clean up the memory that was needed by the class

8.2.3 Member Function Documentation

8.2.3.1 int Collector::are_resIDs_unique (std::vector< ResourceData * >, inFilePosition &, char **) [private]

Check for unique resource IDs.

This method checks if there are all resource IDs unique and returns a warning if they are not

Parameters

in	<i>p_entries</i>	List of resource data (of type resEntry)
out	<i>p_doubleIDPos</i>	Position of double ID
out	<i>p_doubleID</i>	Resource ID that is not unique

Return values

<i>NO_ERROR</i>	All IDs are unique
<i>WARNING_DOUBLE_RESOURCE_ID</i>	A resource ID appears more than once
<i>ERROR_INVALID_PARAMETER</i>	One or more parameters are invalid

Remarks

The caller is responsible to free the string containing the name of the resource ID that appears more than once

8.2.3.2 `int Collector::collect (std::vector< ResourceData * > *, Irc::CompressionType , Irc::EncryptionType , const unsigned char *)`

Collect, compress and encrypt data.

This method collects all resource data files, reads them into memory, compresses and encrypts it if desired and finally generates the one big resource file.

Parameters

in	<i>p_resEntries</i>	Collection of data resource entries
in	<i>p_compress</i>	Compression type for complete file
in	<i>p_encrypt</i>	Encryption type for complete file
in	<i>p_key</i>	Password if encryption is requested

Return values

<i>NO_ERROR</i>	Resource Data File successfully compiled
<i>ERROR_FILE_OPEN</i>	An error occurred opening the rdf file
<i>ERROR_FILE_NOT_FOUND</i>	The desired resource data file could not be found
<i>ERROR_INVALID_PARAMETER</i>	The provided parameter was nullptr
<i>ERROR_FILE_NOT_FOUND</i>	The resource file could not be found
<i>ERROR_COMPRESSION_NOT_AVAILABLE</i>	The selected compression is not available
<i>ERROR_COMPRESSION_COMPRESS</i>	An error occurred while compressing the complete file
<i>ERROR_ENCRYPTION_NOT_AVAILABLE</i>	The selected encryption is not available
<i>ERROR_ENCRYPTION_ENCRYPT</i>	An error occurred while encrypting the complete file
<i>ERROR_FILE_READ</i>	An error occurred while reading the file
<i>ERROR_FILE_WRITE</i>	An error occurred while writing the file

8.2.3.3 `void Collector::show_resource_data_error (int , ResourceData *) [private]`

Show error message from resource data.

This method shows an appropriate error message if the return value from the [ResourceData](#) class indicates an error

Parameters

in	<i>p_errorCode</i>	Error code
in	<i>p_resData</i>	ResourceData instance that caused the error

8.2.4 Member Data Documentation

8.2.4.1 `char* Collector::m_rcName [private]`

Filename of .rc file.

Definition at line 50 of file Collector.hxx.

8.2.4.2 char* Collector::m_rdfName [private]

Filename for .rdf file.

Definition at line 51 of file Collector.hxx.

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

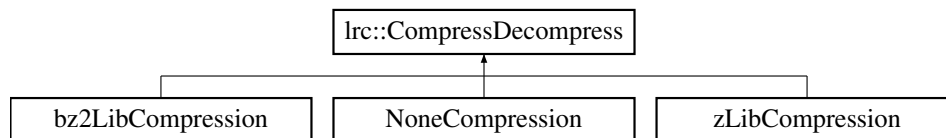
- /home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx

8.3 lrc::CompressDecompress Class Reference

Compression and Decompression base class.

```
#include <CompressDecompress.hxx>
```

Inheritance diagram for lrc::CompressDecompress:



Public Member Functions

- virtual int [compress](#) (const unsigned char *, size_t, unsigned char **, size_t &)=0
Abstract method for compression.
- virtual int [decompress](#) (const unsigned char *, size_t, unsigned char **, size_t &)=0
Abstract method for decompression.

8.3.1 Detailed Description

Compression and Decompression base class.

The class [CompressDecompress](#) is the base class for all compression and decompression in lrc and liblrc.

Definition at line 57 of file CompressDecompress.hxx.

8.3.2 Member Function Documentation

8.3.2.1 virtual int lrc::CompressDecompress::compress (const unsigned char *, size_t, unsigned char **, size_t &) [pure virtual]

Abstract method for compression.

This abstract method has to be implemented by a derived class for compression

Parameters

in	<i>p_decompData</i>	Not yet compressed resource data
in	<i>p_decompSize</i>	Size of the not yet compressed resource data
out	<i>p_compData</i>	Compressed resource data
out	<i>p_compSize</i>	Size of the compressed resource data

Return values

<code>NO_ERROR</code>	Data successfully compressed
-----------------------	------------------------------

Remarks

The caller is responsible to free the used memory

Implemented in [NoneCompression](#), [bz2LibCompression](#), and [zLibCompression](#).

```
8.3.2.2 virtual int Irc::CompressDecompress::decompress ( const unsigned char *, size_t, unsigned char **, size_t & )
           [pure virtual]
```

Abstract method for decompression.

This abstract method has to be implemented by a derived class for decompression

Parameters

in	<code>p_compData</code>	Compressed resource data
in	<code>p_compSize</code>	Size of compressed resource data
out	<code>p_decompData</code>	Decompressed resource data
out	<code>p_decompSize</code>	Size of decompressed resource data

Return values

<code>NO_ERROR</code>	Data successfully decompressed
-----------------------	--------------------------------

Remarks

The caller is responsible to free the used memory

Implemented in [NoneCompression](#), [bz2LibCompression](#), and [zLibCompression](#).

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

- `/home/andy/Programming/Projects/Irc/src/include/CompressDecompress.hxx`

8.4 CompressionFactory Class Reference

Factory to create compression class instances.

```
#include <Factories.hxx>
```

Static Public Member Functions

- static `Irc::CompressDecompress *` `get_compression_class (Irc::CompressionType)`
Creates an instance of the desired class.

8.4.1 Detailed Description

Factory to create compression class instances.

This class is used to create an instance of a `Irc::CompressDecompress` class. The created class instance compresses and decompresses the data, depending on the given `Irc::CompressionType`

Definition at line 48 of file `Factories.hxx`.

8.4.2 Member Function Documentation

8.4.2.1 static lrc::CompressDecompress* CompressionFactory::get_compression_class (lrc::CompressionType) [static]

Creates an instance of the desired class.

This method returns an instance of the desired compression/decompression class (if possible) or `nullptr`

Parameters

in	<i>p_compType</i>	Compression type
----	-------------------	------------------

Returns

Instance of desired compression class

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

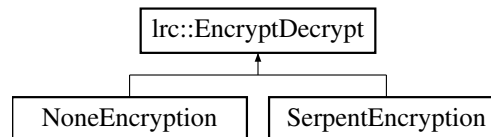
- [/home/andy/Programming/Projects/lrc/src/Factories.hxx](#)

8.5 lrc::EncryptDecrypt Class Reference

Encryption and Decryption base class.

```
#include <EncryptDecrypt.hxx>
```

Inheritance diagram for lrc::EncryptDecrypt:



Public Member Functions

- virtual int [encrypt](#) (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)=0
Abstract method for encryption.
- virtual int [decrypt](#) (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)=0
Abstract method for decryption.

8.5.1 Detailed Description

Encryption and Decryption base class.

This class is the base class for all encryption and decryption in `lrc` and `liblrc`

Definition at line 56 of file `EncryptDecrypt.hxx`.

8.5.2 Member Function Documentation

8.5.2.1 virtual int lrc::EncryptDecrypt::decrypt (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &) [pure virtual]

Abstract method for decryption.

This abstract method has to be implemented by a derived class for decryption

Parameters

in	<i>p_key</i>	Key for decryption
in	<i>p_encData</i>	Encrypted resource data for decryption
in	<i>p_encSize</i>	Size of encrypted resource data
out	<i>p_clearData</i>	Decrypted/clear resource data
out	<i>p_clearSize</i>	Size of clear resource data

Return values

<i>NO_ERROR</i>	Resource data successfully decrypted
<i>ERROR_INVALID_PARAMETER</i>	The return buffer <i>p_clearData</i> was <code>nullptr</code>

Remarks

The caller is responsible to free the used memory

Implemented in [SerpentEncryption](#), and [NoneEncryption](#).

```
8.5.2.2 virtual int Irc::EncryptDecrypt::encrypt ( const unsigned char *, const unsigned char *, size_t, unsigned char **,
size_t & ) [pure virtual]
```

Abstract method for encryption.

This abstract method has to be implemented by a derived class for encryption

Parameters

in	<i>p_key</i>	Key for encryption
in	<i>p_clearData</i>	Clear resource data for encryption
in	<i>p_clearSize</i>	Size of clear resource data
out	<i>p_encData</i>	Encrypted resource data
out	<i>p_encSize</i>	Size of encrypted data

Return values

<i>NO_ERROR</i>	Resource data successfully encrypted
<i>ERROR_INVALID_PARAMETER</i>	The return buffer <i>p_encData</i> was <code>nullptr</code>

Remarks

The caller is responsible to free the used memory

Implemented in [SerpentEncryption](#), and [NoneEncryption](#).

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

- `/home/andy/Programming/Projects/Irc/src/include/EncryptDecrypt.hxx`

8.6 EncryptionFactory Class Reference

Factory to create encryption class instances.

```
#include <Factories.hxx>
```


Static Public Member Functions

- static `Irc::EncryptDecrypt * get_encryption_class (Irc::EncryptionType, char *)` throw (`IrcEncryptionDisabledException`)

Creates an instance of the desired class.

8.6.1 Detailed Description

Factory to create encryption class instances.

This class is used to create an instance of a `Irc::EncryptDecrypt` class. The created class instance encrypts and decrypts the data, depending on the given `Irc::EncryptionType`

Definition at line 70 of file `Factories.hxx`.

8.6.2 Member Function Documentation

8.6.2.1 `static Irc::EncryptDecrypt* EncryptionFactory::get_encryption_class (Irc::EncryptionType , char *)` throw (`IrcEncryptionDisabledException`) `[static]`

Creates an instance of the desired class.

This method returns an instance of the desired encryption/decryption class (if possible of `nullptr` otherwise

Parameters

in	<code>p_encType</code>	Encryption type
in	<code>p_resID</code>	ID of resource that requests encryption

Returns

Instance of desired encryption class

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

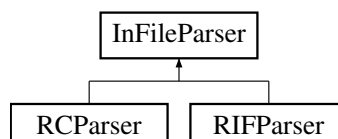
- `/home/andy/Programming/Projects/lrc/src/Factories.hxx`

8.7 InFileParser Class Reference

Base class for all `lrc` input files.

```
#include <InFileParser.hxx>
```

Inheritance diagram for `InFileParser`:



Public Member Functions

- `InFileParser (char *)` throw (`IrcFileNotFoundException`)

Constructor.

- [~InFileParser](#) (void)
Destructor.
- virtual int [parse](#) (void)=0
Parses the file.
- virtual int [get_internal_error](#) ([inFilePosition](#) &, char **)
Returns internal error.
- std::vector< [ResourceData](#) * > * [get_resource_entries](#) (void)
Return all resource entries.

Static Public Member Functions

- static [Irc::CompressionType](#) [eval_compression_type](#) (const char *)
Evaluate compression type from string.
- static [Irc::EncryptionType](#) [eval_encryption_type](#) (const char *)
Evaluate encryption type from string.
- static unsigned char * [get_password](#) (const char *) throw ([IrcFileNotFoundException](#))
Get password.

Protected Types

- enum [internalErrorType](#) {
[ieNone](#), [ieInvalidElement](#), [ieldentNotFound](#), [ieMissingPassword](#),
[ieFilenameNotFound](#), [iePasswordFileNotFound](#) }
Possible internal errors.

Protected Member Functions

- virtual void [clear_internal_error](#) (void)
Clear all internal errors.

Protected Attributes

- char * [m_filename](#)
Filename of the RC file.
- std::vector< [ResourceData](#) * > * [m_resEntries](#)
List of resource data entries.
- [inFilePosition](#) [m_errorPosition](#)
Line and column of error.
- int [m_lastError](#)
Error code of last error.
- [internalErrorType](#) [m_internalError](#)
Internal error.

8.7.1 Detailed Description

Base class for all `lrc` input files.

This class is the base class for all input files for the Linux Resource Compiler. All input file parser must derive from it.

Definition at line 47 of file `InFileParser.hxx`.

8.7.2 Member Enumeration Documentation

8.7.2.1 enum InFileParser::internalErrorType [protected]

Possible internal errors.

An enumeration of all possible internal errors

Enumerator:

ieNone No internal error.

ieInvalidElement The root or any other element has a wrong name.

ieIdentNotFound The identifier could not be found.

ieMissingPassword An encryption is used, but no password is given.

ieFilenameNotFound The actual resource filename is missing.

iePasswordFileNotFound The password file could not be found.

Definition at line 54 of file InFileParser.hxx.

8.7.3 Constructor & Destructor Documentation

8.7.3.1 InFileParser::InFileParser (char *) throw (IrcFileNotFoundException)

Constructor.

This is the constructor. It expects the name of the file to parse and initializes all the members of the class

Parameters

in	p_filename	Filename of input file
----	------------	------------------------

Exceptions

IrcFileNotFoundException	Exception that is thrown if the given file could not be found
--	---

8.7.3.2 InFileParser::~InFileParser (void)

Destructor.

Cleans up the memory and resources the parser class needed

Exceptions

IrcFileNotFoundException	This exception will be thrown if the given file for parsing could not be found
--	--

8.7.4 Member Function Documentation

8.7.4.1 virtual void InFileParser::clear_internal_error (void) [protected], [virtual]

Clear all internal errors.

This method is used to clear all internal errors. It is virtual and should be overwritten by derived classes

8.7.4.2 static `Irc::CompressionType InFileParser::eval_compression_type (const char *) [static]`

Evaluate compression type from string.

This method evaluates which compression type should be used, defined by the given string

Parameters

in	<i>p_compStr</i>	Compression type as string
----	------------------	----------------------------

Return values

<i>Irc::NoneCompression</i>	No compression
<i>Irc::zLibCompression</i>	zlib compression

Remarks

[*Irc::NoneCompression*](#) is the default if the given string defines no other compression type

8.7.4.3 static `Irc::EncryptionType InFileParser::eval_encryption_type (const char *) [static]`

Evaluate encryption type from string.

This method evaluated which encryption type should be used defined by the given string

Parameters

in	<i>p_encStr</i>	Encryption type as string
----	-----------------	---------------------------

Return values

<i>Irc::NoneEncryption</i>	No encryption
<i>Irc::SerpentEncryption</i>	Serpent encryption

Remarks

[*Irc::NoneEncryption*](#) is the default if the givne string defines no other encryption type

8.7.4.4 virtual `int InFileParser::get_internal_error (inFilePosition &, char **) [virtual]`

Returns internal error.

This method returns the state of the last internal error together with the line and column position where it happened and an error message

Remarks

The internal error state will be cleared after that call

Parameters

out	<i>p_errPos</i>	Line and column of error
out	<i>p_errMsg</i>	Error message

Returns

Error code of last error

8.7.4.5 `static unsigned char* InFileParser::get_password (const char *) throw (IrcFileNotFoundException)`
`[static]`

Get password.

This method gets the password, either directly from the .rc or .rif file or it is getting the password from a different file if the "@" notation (re-direct to a file) is used.

Parameters

<code>in</code>	<code>p_passwdStr</code>	Password string
-----------------	--------------------------	-----------------

Returns

Password directly from input file or from re-directed file

Exceptions

IrcFileNotFoundException	This exception will be thrown if the password should be read from a file and the given file could not be found
--	--

Remarks

The caller is responsible to free the used memory that the returned password needs

8.7.4.6 `std::vector<ResourceData *>* InFileParser::get_resource_entries (void)`

Return all resource entries.

This method returns all parsed resource entries in a vector of pointers to a [ResourceData](#) class

Returns

Resource entries

8.7.4.7 `virtual int InFileParser::parse (void) [pure virtual]`

Parses the file.

This method parses the file, creates a [ResourceData](#) class for each entry and adds them to the internal structure

Remarks

If the method returns `ERROR_PARSE`, `get_internal_error` will provide more information

Return values

<code>NO_ERROR</code>	File successfully parsed
<code>ERROR_FILE_OPEN</code>	An error occurred while trying to open the file
<code>ERROR_PARSE</code>	An error occurred while trying to parse the file

Implemented in [RCParser](#), and [RIFParser](#).

8.7.5 Member Data Documentation

8.7.5.1 inFilePosition InFileParser::m_errorPosition [protected]

Line and column of error.

Definition at line 65 of file InFileParser.hxx.

8.7.5.2 char* InFileParser::m_filename [protected]

Filename of the RC file.

Definition at line 63 of file InFileParser.hxx.

8.7.5.3 internalErrorType InFileParser::m_internalError [protected]

Internal error.

Definition at line 67 of file InFileParser.hxx.

8.7.5.4 int InFileParser::m_lastError [protected]

Error code of last error.

Definition at line 66 of file InFileParser.hxx.

8.7.5.5 std::vector<ResourceData *>* InFileParser::m_resEntries [protected]

List of resource data entries.

Definition at line 64 of file InFileParser.hxx.

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

- [/home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx](#)

8.8 IrcEncryptionDisabledException Class Reference

Exception if encryption is disabled but required.

```
#include <lrcExceptions.hxx>
```

Public Member Functions

- [IrcEncryptionDisabledException](#) (char *)
Constructor.
- [~IrcEncryptionDisabledException](#) (void) throw ()
Destructor.
- virtual const char * [what](#) (void) const throw ()
Gives a reason for the exception.

Private Attributes

- char * `m_resourceID`
Resource that requires encryption.

8.8.1 Detailed Description

Exception if encryption is disabled but required.

This exception is thrown if the compiler is compiled without encryption support, but the defined input file (.rc or .rif) defines to encrypt a resource

Definition at line 111 of file IrcExceptions.hxx.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 IrcEncryptionDisabledException::IrcEncryptionDisabledException (char *)

Constructor.

The constructor expects the name of the resource that requires encryption

Parameters

in	<i>p_resourceID</i>	Resource ID
----	---------------------	-------------

8.8.2.2 IrcEncryptionDisabledException::~IrcEncryptionDisabledException (void) throw ()

Destructor.

Frees the memory of the exception

8.8.3 Member Function Documentation

8.8.3.1 virtual const char* IrcEncryptionDisabledException::what (void) const throw () [virtual]

Gives a reason for the exception.

Returns a message explaining that the user disabled encryption at compile time and therefore encryption of the resource is not possible

Returns

- Reason for exception

8.8.4 Member Data Documentation

8.8.4.1 char* IrcEncryptionDisabledException::m_resourceID [private]

Resource that requires encryption.

Definition at line 114 of file IrcExceptions.hxx.

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

- [/home/andy/Programming/Projects/lrc/src/IrcExceptions.hxx](#)

8.9 IrcFileExistsException Class Reference

Exception if an existing file should be overwritten.

```
#include <IrcExceptions.hxx>
```

Public Member Functions

- [IrcFileExistsException](#) (char *)
Constructor.
- [~IrcFileExistsException](#) (void) throw ()
Destructor.
- virtual const char * [what](#) (void) const throw ()
Gives a reason for the exception.

Private Attributes

- char * [m_fileOverwrite](#)
Filename of file that should have been overwritten.

8.9.1 Detailed Description

Exception if an existing file should be overwritten.

This exception is thrown if an existing file should be overwritten, but overwriting is not allowed

Definition at line 75 of file IrcExceptions.hxx.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 IrcFileExistsException::IrcFileExistsException (char *)

Constructor.

The constructor expects the filename of the file that should be overwritten

Parameters

<code>in</code>	<code>p_filename</code>	File of file to be overwritten
-----------------	-------------------------	--------------------------------

8.9.2.2 IrcFileExistsException::~IrcFileExistsException (void) throw ()

Destructor.

Frees up memory needed by the class

8.9.3 Member Function Documentation

8.9.3.1 virtual const char* IrcFileExistsException::what (void) const throw () [virtual]

Gives a reason for the exception.

Returns the reason the exception

Returns

Reason for exception

8.9.4 Member Data Documentation**8.9.4.1** `char* IrcFileExistsException::m_fileOverwrite` `[private]`

Filename of file that should have been overwritten.

Definition at line 78 of file IrcExceptions.hxx.

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

- [/home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx](#)

8.10 IrcFileNotFoundException Class Reference

Exception class if a file could not be found.

```
#include <lrcExceptions.hxx>
```

Public Member Functions

- [IrcFileNotFoundException](#) (char *)
Constructor.
- [~IrcFileNotFoundException](#) (void) throw ()
Destructor.
- virtual const char * [what](#) (void) const throw ()
Method to return reason.

Private Attributes

- char * [m_fileNotFound](#)
Filename of file that could not be found.

8.10.1 Detailed Description

Exception class if a file could not be found.

This is the exception that gets thrown if a file could not be found

Definition at line 41 of file IrcExceptions.hxx.

8.10.2 Constructor & Destructor Documentation**8.10.2.1** `IrcFileNotFoundException::IrcFileNotFoundException (char *)`

Constructor.

The constructor expects the filename of the file that could not be found

Parameters

in	<i>p_fileNotFound-Name</i>	Name of file that could not be found
----	----------------------------	--------------------------------------

8.10.2.2 IrcFileNotFoundException::~~IrcFileNotFoundException (void) throw ()

Destructor.

8.10.3 Member Function Documentation

8.10.3.1 virtual const char* IrcFileNotFoundException::what (void) const throw () [virtual]

Method to return reason.

The overwritten method what is used to return the reason of the exception

Returns

Reason of exception

8.10.4 Member Data Documentation

8.10.4.1 char* IrcFileNotFoundException::m_fileNotFound [private]

Filename of file that could not be found.

Definition at line 44 of file IrcExceptions.hxx.

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

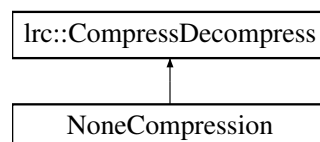
- [/home/andy/Programming/Projects/Irc/src/IrcExceptions.hxx](#)

8.11 NoneCompression Class Reference

Compression class that does *NO* compression.

```
#include <NoneCompression.hxx>
```

Inheritance diagram for NoneCompression:



Public Member Functions

- int [compress](#) (const unsigned char *, size_t, unsigned char **, size_t &)
No compression.
- int [decompress](#) (const unsigned char *, size_t, unsigned char **, size_t &)
No decompression.

8.11.1 Detailed Description

Compression class that does *NO* compression.

This class does no compression at all. It is used for the simplest case of no compression. It is created and implemented nonetheless to fit in the Strategy Pattern

Definition at line 46 of file NoneCompression.hxx.

8.11.2 Member Function Documentation

8.11.2.1 `int NoneCompression::compress (const unsigned char *, size_t, unsigned char **, size_t &) [virtual]`

No compression.

This method does no compression and returns a copy of the given data.

Implements [Irc::CompressDecompress](#).

8.11.2.2 `int NoneCompression::decompress (const unsigned char *, size_t, unsigned char **, size_t &) [virtual]`

No decompression.

This method does no decompression and returns a copy of the given data.

Implements [Irc::CompressDecompress](#).

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

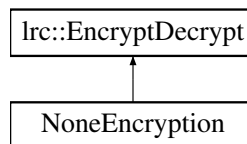
- [/home/andy/Programming/Projects/Irc/src/strategies/NoneCompression.hxx](#)

8.12 NoneEncryption Class Reference

Encryption class that does *NO* encryption.

```
#include <NoneEncryption.hxx>
```

Inheritance diagram for NoneEncryption:



Public Member Functions

- `int encrypt (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)`
No encryption.
- `int decrypt (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)`
No decryption.

8.12.1 Detailed Description

Encryption class that does *NO* encryption.

This class does no encryption at all. It is used for the simplest case of no encryption. It is created and implemented nonetheless to fit in the Strategy Pattern.

Definition at line 45 of file NoneEncryption.hxx.

8.12.2 Member Function Documentation

8.12.2.1 `int NoneEncryption::decrypt (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)`
 [virtual]

No decryption.

This method does not decryption and returns a copy of the given data

Implements [Irc::EncryptDecrypt](#).

8.12.2.2 `int NoneEncryption::encrypt (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)`
 [virtual]

No encryption.

This method does no encryption and returns a copy of the given data

Implements [Irc::EncryptDecrypt](#).

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

- [/home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx](#)

8.13 ParserFactory Class Reference

Factory class to create an appropriate parser class.

```
#include <ParserFactory.hxx>
```

Static Public Member Functions

- static [InFileParser](#) * [create_input_parser](#) (const char *)
Creates an appropriate parser class.

8.13.1 Detailed Description

Factory class to create an appropriate parser class.

This factory class is used to create a parser class that is able to parse a given input file. There are two possibilities: Parsing a .rc file or parsing a .rif file (xml).

Definition at line 46 of file ParserFactory.hxx.

8.13.2 Member Function Documentation

8.13.2.1 `static InFileParser* ParserFactory::create_input_parser (const char *)` [static]

Creates an appropriate parser class.

This method creates a parser class depending on the given file. The file can be a .rc or a .rif (xml) file

Parameters

<code>in</code>	<code>p_filename</code>	Name of the file to parse
-----------------	-------------------------	---------------------------

Returns

Instance of a matching parser class

Remarks

The caller is responsible to free/delete the returned class
 If the file does not exist `nullptr` will be returned

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

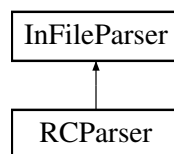
- `/home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx`

8.14 RCParse Class Reference

Class to parse an .rc file.

```
#include <RCParse.hxx>
```

Inheritance diagram for RCParse:

**Public Member Functions**

- `RCParse` (`char *`) `throw (IrcFileNotFoundException)`
Constructor.
- `int parse` (`void`)
Parses the file.

Private Member Functions

- `bool is_comment` (`char *`)
Checks if the line is a comment line.

Additional Inherited Members**8.14.1 Detailed Description**

Class to parse an .rc file.

The `RCParse` class is responsible to parse a RC file. The RC file format is quite simple and somewhat similar to the Windows rc files.

Definition at line 46 of file `RCParse.hxx`.

8.14.2 Constructor & Destructor Documentation**8.14.2.1 RCParse::RCParse (char *) throw (IrcFileNotFoundException)**

Constructor.

This is the only constructor of the class and it expects a filename as parameter

Parameters

<code>in</code>	<code>p_filename</code>	Filename of RC file
-----------------	-------------------------	---------------------

Exceptions

IrcFileNotFoundException	Exception that is thrown if the given .rc file could not be found
--	---

8.14.3 Member Function Documentation

8.14.3.1 `bool RCParse::is_comment (char *) [private]`

Checks if the line is a comment line.

This methods reads the given text as a text line from left to right abd checks if there are any characters at all and if there are, if they start with a '#'. An empty line or a line starting with '#' is treated as comment line

Parameters

<code>in</code>	<code>p_line</code>	Line of text
-----------------	---------------------	--------------

Return values

<code>true</code>	Line is comment or empty
<code>false</code>	Line contains information

8.14.3.2 `int RCParse::parse (void) [virtual]`

Parses the file.

This method parses the .rc file, creates a [ResourceData](#) class for each entry and adds them to the internal structure

Remarks

If the method returns `ERROR_PARSE`, `get_internal_error` will provide more information

Return values

<code>NO_ERROR</code>	File successfully parsed
<code>ERROR_FILE_OPEN</code>	An error occurred while trying to open the file
<code>ERROR_PARSE</code>	An error occurred while trying to parse the file

Implements [InFileParser](#).

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

- `/home/andy/Programming/Projects/lrc/src/compiler/RCParse.hxx`

8.15 `resEntry_ Struct Reference`

Data for one resource entry.

```
#include <Resource.hxx>
```

Public Attributes

- char [resID](#) [[MAX_ID_LEN](#)]
Resource ID.
- unsigned int [startOffset](#)
Offset from the start.
- unsigned int [resSize](#)
Size of resource.
- [Irc::EncryptionType](#) [encType](#)
Type of encryption of this entry.
- [Irc::CompressionType](#) [compType](#)
Type of compression of this entry.

8.15.1 Detailed Description

Data for one resource entry.

This struct contains the data for one resource entry in a form to save to the .rdf file

Definition at line 88 of file Resource.hxx.

8.15.2 Member Data Documentation

8.15.2.1 [Irc::CompressionType](#) [resEntry_::compType](#)

Type of compression of this entry.

Definition at line 93 of file Resource.hxx.

8.15.2.2 [Irc::EncryptionType](#) [resEntry_::encType](#)

Type of encryption of this entry.

Definition at line 92 of file Resource.hxx.

8.15.2.3 char [resEntry_::resID](#)[[MAX_ID_LEN](#)]

Resource ID.

Definition at line 89 of file Resource.hxx.

8.15.2.4 unsigned int [resEntry_::resSize](#)

Size of resource.

Definition at line 91 of file Resource.hxx.

8.15.2.5 unsigned int [resEntry_::startOffset](#)

Offset from the start.

Definition at line 90 of file Resource.hxx.

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

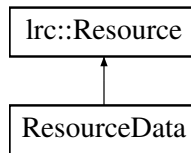
- [/home/andy/Programming/Projects/Irc/src/include/Resource.hxx](#)

8.16 Irc::Resource Class Reference

[Resource](#) class for use with library.

```
#include <Resource.hxx>
```

Inheritance diagram for Irc::Resource:



Public Member Functions

- [Resource](#) (void)
Constructor.
- [~Resource](#) (void)
Destructor.
- char * [get_ID](#) (void)
Returns the ID of the resource.
- unsigned char * [get_res_data](#) (void)
Actual resource data.
- size_t [get_res_size](#) (void)
Actual resource size.

Protected Attributes

- char * [m_resID](#)
Resource ID.
- unsigned char * [m_resData](#)
Pointer to resource data.
- size_t [m_resSize](#)
Size of resource data.

8.16.1 Detailed Description

[Resource](#) class for use with library.

This class is used to define one resource for the developer that uses Irc. It contains a pointer to the data and the size of the data. It also provides its ID.

Definition at line 122 of file Resource.hxx.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 Irc::Resource::Resource (void)

Constructor.

This is a standard constructor. It initializes all members to default values

Remarks

The values will be filled by the derived class

8.16.2.2 Irc::Resource::~~Resource (void)

Destructor.

Cleans up the used memory of the class

8.16.3 Member Function Documentation

8.16.3.1 char* Irc::Resource::get_ID (void)

Returns the ID of the resource.

Method to return the ID of the resource

Returns

ID of resource

8.16.3.2 unsigned char* Irc::Resource::get_res_data (void)

Actual resource data.

This method returns a pointer to the actual resource data as bytes

Returns

Pointer to resource data

8.16.3.3 size_t Irc::Resource::get_res_size (void)

Actual resource size.

Method to return the actual size of the resource (in bytes)

Returns

Size of resource

8.16.4 Member Data Documentation

8.16.4.1 unsigned char* Irc::Resource::m_resData [protected]

Pointer to resource data.

Definition at line 126 of file Resource.hxx.

8.16.4.2 char* Irc::Resource::m_resID [protected]

[Resource ID](#).

Definition at line 125 of file Resource.hxx.

8.16.4.3 `size_t Irc::Resource::m_resSize` [protected]

Size of resource data.

Definition at line 127 of file Resource.hxx.

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

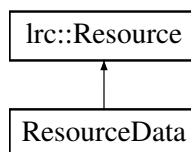
- </home/andy/Programming/Projects/lrc/src/include/Resource.hxx>

8.17 ResourceData Class Reference

Internal class for resource with more information.

```
#include <ResourceData.hxx>
```

Inheritance diagram for ResourceData:



Public Member Functions

- [ResourceData](#) (void)
Constructor.
- [~ResourceData](#) (void)
Destructor.
- void [set_ident](#) (const char *)
Set resource identifier.
- void [set_file](#) (const char *)
Set file containing resource.
- char * [get_file](#) (void)
Return filename.
- void [set_encryption](#) ([Irc::EncryptionType](#), const unsigned char *)
Set encryption type.
- [Irc::EncryptionType](#) [get_encryption](#) (void)
Return encryption type.
- void [set_compression](#) ([Irc::CompressionType](#))
Set compression type.
- [Irc::CompressionType](#) [get_compression](#) (void)
Return compression type.
- void [set_rc_position](#) (int, int)
Set position of resource description.
- [inFilePosition](#) [get_rc_position](#) (void)
Return resource description position.
- int [prepare_resource_from_file](#) (unsigned char **, [size_t](#) &)
Prepare resource for collecting.
- int [get_data_from_memory](#) (unsigned char *, [resEntry](#), const unsigned char *p_password=nullptr)
Get resource data from compressed and encrypted chunk of memory.
- char * [get_error_msg](#) (void)
Return error message.

Protected Member Functions

- void [set_error_msg](#) (char *)
Set new error message.

Protected Attributes

- char * [m_filename](#)
Filename of file containing resource.
- [Irc::EncryptionType](#) [m_encryption](#)
Type of encryption for this resource.
- [Irc::CompressionType](#) [m_compression](#)
Type of compression for this resource.
- [inFilePosition](#) [m_inFilePosition](#)
Position of resource in RC file.
- char * [m_errorMsg](#)
Error message (if any)

Private Attributes

- unsigned char * [m_password](#)
Password for encryption.

8.17.1 Detailed Description

Internal class for resource with more information.

This class contains all needed data of a resource. It is derived from Resource
Definition at line 53 of file ResourceData.hxx.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 ResourceData::ResourceData (void)

Constructor.

Standard constructor to initialize the class

8.17.2.2 ResourceData::~ResourceData (void)

Destructor.

Cleans up memory used from the class

8.17.3 Member Function Documentation

8.17.3.1 Irc::CompressionType ResourceData::get_compression (void)

Return compression type.

Method to return the compression type of this very resource

Returns

Compression type of resource

8.17.3.2 `int ResourceData::get_data_from_memory (unsigned char * , resEntry , const unsigned char * p_password = nullptr)`

Get resource data from compressed and encrypted chunk of memory.

This method extracts the information of the class from a compressed and encrypted chunk of memory. It is usually used after the data is loaded from file

Parameters

in	<i>p_dataStart</i>	Start adress of memory block
in	<i>p_resEntry</i>	Entry block of resource
in	<i>p_password</i>	Password if resource is encrypted

Return values

<i>NO_ERROR</i>	Resource data successfully extracted
<i>ERROR_INVALID_PARAMETER</i>	One or more provided parameters
<i>ERROR_COMPRESSION_NOT_AVAILABLE</i>	The required compression class is not available
<i>ERROR_ENCRYPTION_NOT_AVAILABLE</i>	The required encryption class is not available
<i>ERROR_ENCRYPTION_DECRYPT</i>	An error occurred while decrypting the data
<i>ERROR_COMPRESSION_DECOMPRESS</i>	An error occurred while decompressing the data

Remarks

The caller is responsible to free the returned class

8.17.3.3 `Irc::EncryptionType ResourceData::get_encryption (void)`

Return encryption type.

Method to return the encryption type of this very resource

Returns

Encryption type of resource

8.17.3.4 `char* ResourceData::get_error_msg (void)`

Return error message.

This method returns the internal error message (if there is any) and clears it

Returns

Internal error message

Remarks

The caller is responsible to free the used memory for the message

8.17.3.5 char* ResourceData::get_file (void)

Return filename.

This method returns the filename of the resource data file of this very resource

Returns

Filename of resource data file

8.17.3.6 inFilePosition ResourceData::get_rc_position (void)

Return resource description position.

Method to return the resource description in the RC file of this very resource

Returns

Resource position in RC file

8.17.3.7 int ResourceData::prepare_resource_from_file (unsigned char **, size_t &)

Prepare resource for collecting.

This method loads the file containing the resource, compresses and encrypts the data and provides it for the collectors

Parameters

out	<i>p_resData</i>	Resource data for collector
out	<i>p_resSize</i>	Size of resource data

Return values

<i>NO_ERROR</i>	Resource successfully prepared
<i>ERROR_INVALID_PARAMETER</i>	The provided parameter was nullptr
<i>ERROR_FILE_NOT_FOUND</i>	The resource file could not be found
<i>ERROR_COMPRESSION_NOT_AVAILABLE</i>	The selected compression is not available
<i>ERROR_ENCRYPTION_NOT_AVAILABLE</i>	The selected encryption is not available
<i>ERROR_FILE_OPEN</i>	The file could not be opened
<i>ERROR_FILE_READ</i>	An error occurred while reading the file

Remarks

The caller is responsible to free the used memory

8.17.3.8 void ResourceData::set_compression (Irc::CompressionType)

Set compression type.

Method to set compression type of resource

Parameters

in	<i>p_res- Compression</i>	Compression type
----	-------------------------------	------------------

8.17.3.9 void ResourceData::set_encryption (Irc::EncryptionType , const unsigned char *)

Set encryption type.

Method to set encryption type of resource

Parameters

in	<i>p_resEncryption</i>	Encryption type
in	<i>p_password</i>	Password for encryption

8.17.3.10 void ResourceData::set_error_msg (char *) [protected]

Set new error message.

This method clears an old error message (if any) and sets the new given one

Parameters

in	<i>p_newErrMsg</i>	New error message
----	--------------------	-------------------

8.17.3.11 void ResourceData::set_file (const char *)

Set file containing resource.

Method to set the filename containing the resource data

Parameters

in	<i>p_resFilename</i>	Filename containing resource
----	----------------------	------------------------------

8.17.3.12 void ResourceData::set_ident (const char *)

Set resource identifier.

Method to set the internal resource identifier

Parameters

in	<i>p_resIdent</i>	Resource identifier
----	-------------------	---------------------

8.17.3.13 void ResourceData::set_rc_position (int , int)

Set position of resource description.

Method to set the position of the resource description in the RC file. It is used for a sane error message in the collecting process

Parameters

in	<i>p_line</i>	Line number in RC file
in	<i>p_col</i>	Column number in RC file

See Also

[Collector](#)

8.17.4 Member Data Documentation

8.17.4.1 `Irc::CompressionType ResourceData::m_compression` [protected]

Type of compression for this resource.

Definition at line 61 of file ResourceData.hxx.

8.17.4.2 `Irc::EncryptionType ResourceData::m_encryption` [protected]

Type of encryption for this resource.

Definition at line 60 of file ResourceData.hxx.

8.17.4.3 `char* ResourceData::m_errorMsg` [protected]

Error message (if any)

Definition at line 64 of file ResourceData.hxx.

8.17.4.4 `char* ResourceData::m_filename` [protected]

Filename of file containing resource.

Definition at line 59 of file ResourceData.hxx.

8.17.4.5 `inFilePosition ResourceData::m_inFilePosition` [protected]

Position of resource in RC file.

Definition at line 62 of file ResourceData.hxx.

8.17.4.6 `unsigned char* ResourceData::m_password` [private]

Password for encryption.

Definition at line 56 of file ResourceData.hxx.

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

- [/home/andy/Programming/Projects/lrc/src/ResourceData.hxx](#)

8.18 Irc::ResourceManager Class Reference

Manager class handling all resources of a resource file.

```
#include <ResourceManager.hxx>
```

Public Member Functions

- [ResourceManager](#) (const char *, const [CompressionType](#), const [EncryptionType](#), const unsigned char *)
throw ([IrcFileNotFoundException](#))
Constructor expecting resource file name.
- [~ResourceManager](#) (void)
Destructor.
- char ** [get_resource_ids](#) (int &)
Returns a list of all resources by ID.
- [Resource](#) * [get_resource](#) (const char *, const unsigned char *p_password=nullptr)
Returns the requested resource.
- [Resource](#) * [get_resource](#) (unsigned int, const unsigned char *p_password=nullptr)
Returns the resource at the index.

Private Member Functions

- int [load_from_file](#) (void)
Method to load resource file.

Private Attributes

- char * [m_resourceFile](#)
Filename of resource file.
- [resEntry](#) * [m_resEntries](#)
Entries of the resource file.
- int [m_numResEntries](#)
Number of entries in resource file.
- unsigned char * [m_resData](#)
Pointer to data of resource file.
- size_t [m_resDataSize](#)
Size of resource data.
- [CompressionType](#) [m_compType](#)
Compression type for complete file.
- [EncryptionType](#) [m_encType](#)
Encryption type for complete file.
- unsigned char * [m_password](#)
Password is complete file is encrypted.

8.18.1 Detailed Description

Manager class handling all resources of a resource file.

This is the main class of the `liblrc` library. It expects a filename of a resource file in the constructor and handles all resources in that given file.

Definition at line 51 of file `ResourceManager.hxx`.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 Irc::ResourceManager::ResourceManager (const char * , const CompressionType , const EncryptionType , const unsigned char *) throw (IrcFileNotFoundException)

Constructor expecting resource file name.

The constructor expects the name of a resource file. The file will be loaded at first use, but an exceptions will be thrown if it does not exist

Parameters

in	<i>p_resFilename</i>	Filename of resource file
in	<i>p_compress</i>	Compression type if the complete file is compressed
in	<i>p_encrypt</i>	Encryption type if the complete file is encrypted
in	<i>p_key</i>	Password if whole file is encrypted

Exceptions

IrcFileNotFoundException	Will be thrown if the file does not exist
--	---

8.18.2.2 Irc::ResourceManager::~~ResourceManager (void)

Destructor.

The destructor cleans up all the used memory of the class

8.18.3 Member Function Documentation

8.18.3.1 Resource* Irc::ResourceManager::get_resource (const char * , const unsigned char * p_password = nullptr)

Returns the requested resource.

This method returns the requested resource defined by the given resource ID or `nullptr` if it does not exist

Parameters

in	<i>p_resID</i>	Resource ID
in	<i>p_password</i>	Password if resource is encrypted

Returns

Instance of [Resource](#) class of the requested resource (or `nullptr` if it does not exist)

Remarks

The caller is responsible to free the used memory

8.18.3.2 Resource* Irc::ResourceManager::get_resource (unsigned int, const unsigned char * p_password = nullptr)

Returns the resource at the index.

This method returns the requested resource defined by the given index or `nullptr` if the index is out of range

Parameters

in	<i>p_resIdx</i>	Index of resource
in	<i>p_password</i>	Password if resource is encrypted

Returns

Instance of [Resource](#) class of the resource at the index (or `nullptr` if the index is out of range)

Remarks

Indices are zero based, i.e. valid indices are between 0 and n-1 (n means the number of all resources)
The caller is responsible to free the used memory

8.18.3.3 `char** Irc::ResourceManager::get_resource_ids (int &)`

Returns a list of all resources by ID.

This method returns a list of all resources. The list contains all IDs of the resources

Parameters

out	<i>p_numRes</i>	Number of resources in list
-----	-----------------	-----------------------------

Remarks

The caller is responsible to free the used memory

8.18.3.4 `int Irc::ResourceManager::load_from_file (void) [private]`

Method to load resource file.

This method loads all data from the resource file

Return values

<i>NO_ERROR</i>	Resource file successfully loaded
<i>ERROR_FILE_READ</i>	An error occurred while reading the file

8.18.4 Member Data Documentation**8.18.4.1** `CompressionType Irc::ResourceManager::m_compType [private]`

Compression type for complete file.

Definition at line 59 of file ResourceManager.hxx.

8.18.4.2 `EncryptionType Irc::ResourceManager::m_encType [private]`

Encryption type for complete file.

Definition at line 60 of file ResourceManager.hxx.

8.18.4.3 `int Irc::ResourceManager::m_numResEntries [private]`

Number of entries in resource file.

Definition at line 56 of file ResourceManager.hxx.

8.18.4.4 unsigned char* Irc::ResourceManager::m_password [private]

Password is complete file is encrypted.

Definition at line 61 of file ResourceManager.hxx.

8.18.4.5 unsigned char* Irc::ResourceManager::m_resData [private]

Pointer to data of resource file.

Definition at line 57 of file ResourceManager.hxx.

8.18.4.6 size_t Irc::ResourceManager::m_resDataSize [private]

Size of resource data.

Definition at line 58 of file ResourceManager.hxx.

8.18.4.7 resEntry* Irc::ResourceManager::m_resEntries [private]

Entries of the resource file.

Definition at line 55 of file ResourceManager.hxx.

8.18.4.8 char* Irc::ResourceManager::m_resourceFile [private]

Filename of resource file.

Definition at line 54 of file ResourceManager.hxx.

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

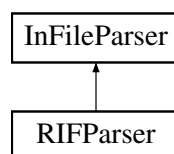
- [/home/andy/Programming/Projects/Irc/src/include/ResourceManager.hxx](#)

8.19 RIFParser Class Reference

Class to parse a .rif (XML) file.

```
#include <RIFParser.hxx>
```

Inheritance diagram for RIFParser:

**Public Member Functions**

- [RIFParser](#) (char *) throw (IrcFileNotFoundException)
Constructor.
- int [parse](#) (void)
Parses the file.

Additional Inherited Members

8.19.1 Detailed Description

Class to parse a .rif (XML) file.

This class is used to parse a .rif file. The .rif file is actually a XML file

Definition at line 44 of file RIFParser.hxx.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 RIFParser::RIFParser (char *) throw (IrcFileNotFoundException)

Constructor.

This is the only constructor of the class and it expects a filename as parameter

Parameters

in	<i>p_filename</i>	Filename of .rif file
----	-------------------	-----------------------

Exceptions

IrcFileNotFoundException	Exception that is thrown if the given .rif file could not be found
--	--

8.19.3 Member Function Documentation

8.19.3.1 int RIFParser::parse (void) [virtual]

Parses the file.

This method parses the .rif file, creates a [ResourceData](#) class for each entry and adds them to the internal structure

Remarks

If the method returns ERROR_PARSE, get_internal_error will provide more information

Return values

<i>NO_ERROR</i>	File successfully parsed
<i>ERROR_FILE_OPEN</i>	An error occurred while trying to open the file
<i>ERROR_PARSE</i>	An error occurred while trying to parse the file

Implements [InFileParser](#).

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

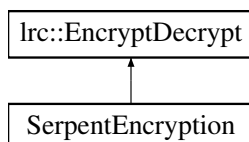
- /home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx

8.20 SerpentEncryption Class Reference

Class to encrypt/decrypt using the *Serpent* algorithm.

```
#include <SerpentEncryption.hxx>
```

Inheritance diagram for SerpentEncryption:



Public Member Functions

- int [encrypt](#) (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)
Serpent encryption.
- int [decrypt](#) (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &)
Serpent decryption.

Protected Member Functions

- int [create_initialization_vector](#) (unsigned char *, size_t)
Create initialization vector.

8.20.1 Detailed Description

Class to encrypt/decrypt using the *Serpent* algorithm.

This class encrypts and decrypts the given data using the *Serpent* algorithm. The algorithm is not implemented here, instead the algorithm from the `crypto++` library is used

Definition at line 47 of file `SerpentEncryption.hxx`.

8.20.2 Member Function Documentation

8.20.2.1 int SerpentEncryption::create_initialization_vector (unsigned char *, size_t) [protected]

Create initialization vector.

This method creates a random initialization vector for the Serpent encryption

Parameters

in, out	<i>p_ivArray</i>	Array for initialization vector
in	<i>p_ivSize</i>	Size of the array

Return values

<i>NO_ERROR</i>	Initialization vector successfully created
<i>ERROR_INVALID_PARAMETER</i>	The array was not created

8.20.2.2 int SerpentEncryption::decrypt (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &) [virtual]

Serpent decryption.

Method to decrypt the given data using the Serpent algorithm

Implements [Irc::EncryptDecrypt](#).

8.20.2.3 `int SerpentEncryption::encrypt (const unsigned char *, const unsigned char *, size_t, unsigned char **, size_t &) [virtual]`

Serpent encryption.

Method to encrypt the given data using the Serpent algorithm

Implements [Irc::EncryptDecrypt](#).

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

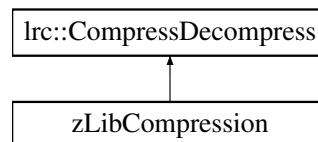
- `/home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx`

8.21 zLibCompression Class Reference

Compression class that uses the *zLib* algorithm.

```
#include <zLibCompression.hxx>
```

Inheritance diagram for zLibCompression:



Public Member Functions

- `int compress (const unsigned char *, size_t, unsigned char **, size_t &)`
zLib compression
- `int decompress (const unsigned char *, size_t, unsigned char **, size_t &)`
zLib decompression

8.21.1 Detailed Description

Compression class that uses the *zLib* algorithm.

This class uses the zLib algorithm for compression and decompression of the resource data.

Definition at line 45 of file zLibCompression.hxx.

8.21.2 Member Function Documentation

8.21.2.1 `int zLibCompression::compress (const unsigned char *, size_t, unsigned char **, size_t &) [virtual]`

zLib compression

This method compresses the given data using the zLib algorithm

Implements [Irc::CompressDecompress](#).

8.21.2.2 `int zLibCompression::decompress (const unsigned char *, size_t, unsigned char **, size_t &) [virtual]`

zLib decompression

This method decompresses the given data using the zLib algorithm

Implements [Irc::CompressDecompress](#).

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

- [/home/andy/Programming/Projects/Irc/src/strategies/zLibCompression.hxx](#)

Chapter 9

File Documentation

9.1 /home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx File Reference

```
#include <vector>
#include "../lrcExceptions.hxx"
#include "../ResourceData.hxx"
```

Classes

- class [Collector](#)
Class to collect all resource data.

9.1.1 Detailed Description

This file contains the class definition of the collector class that is responsible to collect all resource data

Author

Andreas Tschärner

Date

2012-05-06

Definition in file [Collector.hxx](#).

9.2 /home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx File Reference

```
#include <vector>
#include "../ResourceData.hxx"
#include "../lrcExceptions.hxx"
```

Classes

- class [InFileParser](#)
Base class for all lrc input files.

9.2.1 Detailed Description

This file contains the base class for all parsers that read and parse an input file for the Linux Resource Compiler

Author

Andreas Tscharner

Date

2012-09-01

Definition in file [InFileParser.hxx](#).

9.3 /home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx File Reference

```
#include <iostream>
#include <cstring>
#include <unistd.h>
#include "include/CompressDecompress.hxx"
#include "include/EncryptDecrypt.hxx"
#include "../lrcExceptions.hxx"
#include "../Utils.hxx"
#include "InFileParser.hxx"
#include "Collector.hxx"
#include "ParserFactory.hxx"
```

Macros

- `#define VERSION "<undefined>"`
Define for version number if generated header file is missing.

Functions

- void `usage` (int argc, char **argv)
Show the usage of the compiler.
- int `main` (int argc, char **argv)
Main program.

9.3.1 Detailed Description

This is the main file for the compiler. It calls the appropriate parser and then the collector which creates the .rdf file

Author

Andreas Tscharner

Date

2012-05-06

Definition in file [lrc.cxx](#).

9.3.2 Macro Definition Documentation

9.3.2.1 #define VERSION "<undefined>"

Define for version number if generated header file is missing.

Definition at line 45 of file lrc.cxx.

9.3.3 Function Documentation

9.3.3.1 int main (int argc, char ** argv)

Main program.

Definition at line 71 of file lrc.cxx.

9.3.3.2 void usage (int argc, char ** argv)

Show the usage of the compiler.

Definition at line 54 of file lrc.cxx.

9.4 /home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx File Reference

```
#include "InFileParser.hxx"
```

Classes

- class [ParserFactory](#)
Factory class to create an appropriate parser class.

9.4.1 Detailed Description

This file contains the declaration of the [ParserFactory](#) class that is used to create a matching parser class depending on the file to parse.

Author

Andreas Tscherner

Date

2011-09-11

Definition in file [ParserFactory.hxx](#).

9.5 /home/andy/Programming/Projects/lrc/src/compiler/RCParse.hxx File Reference

```
#include <vector>
#include "../lrcExceptions.hxx"
#include "../ResourceData.hxx"
#include "InFileParser.hxx"
```

Classes

- class [RCParser](#)

Class to parse an .rc file.

9.5.1 Detailed Description

This file contains the class definition for the parser for the simple RC format

Author

Andreas Tscharner

Date

2011-09-26

Definition in file [RCParser.hxx](#).

9.6 /home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx File Reference

```
#include "InFileParser.hxx"
```

Classes

- class [RIFParser](#)

Class to parse a .rif (XML) file.

9.6.1 Detailed Description

This file contains the declaration of the [RIFParser](#) class, a class to parse the RIF input file for lrc

Author

Andreas Tscharner

Date

2011-09-26

Definition in file [RIFParser.hxx](#).

9.7 /home/andy/Programming/Projects/lrc/src/Factories.hxx File Reference

```
#include "include/CompressDecompress.hxx"  
#include "include/EncryptDecrypt.hxx"  
#include "lrcExceptions.hxx"
```

Classes

- class [CompressionFactory](#)
Factory to create compression class instances.
- class [EncryptionFactory](#)
Factory to create encryption class instances.

9.7.1 Detailed Description

This file contains the declarations of the compression factory class and the encryption factory class

Author

Andreas Tscharner

Date

2012-01-07

Definition in file [Factories.hxx](#).

9.8 /home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx File Reference

```
#include <stddef.h>
```

Classes

- class [lrc::CompressDecompress](#)
Compression and Decompression base class.

Namespaces

- namespace [lrc](#)
Namespace lrc for classes in the library and for extending lrc.

Enumerations

- enum [lrc::CompressionType](#) { [lrc::NoneCompression](#), [lrc::zLibCompression](#), [lrc::bz2LibCompression](#) }

9.8.1 Detailed Description

File that contains the CompressDecompress class, the base class for all compression/decompression algorithms used in `lrc`

Author

Andreas Tscharner

Date

2012-12-28

Definition in file [CompressDecompress.hxx](#).

9.9 /home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx File Reference

```
#include <stddef.h>
```

Classes

- class [lrc::EncryptDecrypt](#)
Encryption and Decryption base class.

Namespaces

- namespace [lrc](#)
Namespace lrc for classes in the library and for extending lrc.

Enumerations

- enum [lrc::EncryptionType](#) { [lrc::NoneEncryption](#), [lrc::SerpentEncryption](#) }

9.9.1 Detailed Description

This file contains the abstract class `EncryptDecrypt`. It is the base class for all encryption and decryption used in `lrc`.

Author

Andreas Tscharner

Date

2011-08-22

Definition in file [EncryptDecrypt.hxx](#).

9.10 /home/andy/Programming/Projects/lrc/src/include/Resource.hxx File Reference

```
#include <stddef.h>
#include "CompressDecompress.hxx"
#include "EncryptDecrypt.hxx"
```

Classes

- struct [resEntry_](#)
Data for one resource entry.
- class [lrc::Resource](#)
Resource class for use with library.

Namespaces

- namespace [lrc](#)
Namespace lrc for classes in the library and for extending lrc.

Macros

- #define [MAX_ID_LEN](#) 80
Maximum length of resource ID.

Typedefs

- typedef struct [resEntry_ resEntry](#)
Define a better name for struct resEntry_.

9.10.1 Detailed Description

This file contains one of the main classes: Resource

Author

Andreas Tscharner

Date

2012-06-10

Definition in file [Resource.hxx](#).

9.10.2 Macro Definition Documentation

9.10.2.1 #define MAX_ID_LEN 80

Maximum length of resource ID.

Definition at line 80 of file Resource.hxx.

9.10.3 Typedef Documentation

9.10.3.1 typedef struct resEntry_ resEntry

Define a better name for struct [resEntry_](#).

Definition at line 97 of file Resource.hxx.

9.11 /home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx File Reference

```
#include <stddef.h>
#include "Resource.hxx"
#include "../ResourceData.hxx"
#include "../lrcExceptions.hxx"
```

Classes

- class [lrc::ResourceManager](#)
Manager class handling all resources of a resource file.

Namespaces

- namespace [lrc](#)
Namespace lrc for classes in the library and for extending lrc.

9.11.1 Detailed Description

This file contains the declaration of the ResourceManager, the main class for `liblrc`

Author

Andreas Tschärner

Date

2013-01-06

Definition in file [ResourceManager.hxx](#).

9.12 /home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx File Reference

```
#include <exception>
```

Classes

- class [lrcFileNotFoundException](#)
Exception class if a file could not be found.
- class [lrcFileExistsException](#)
Exception if an existing file should be overwritten.
- class [lrcEncryptionDisabledException](#)
Exception if encryption is disabled but required.

9.12.1 Detailed Description

This file contains all exception classes for lrc project

Author

Andreas Tschärner

Date

2012-01-07

Definition in file [lrcExceptions.hxx](#).

9.13 /home/andy/Programming/Projects/lrc/src/ResourceData.hxx File Reference

```
#include <tuple>
#include "include/Resource.hxx"
#include "include/CompressDecompress.hxx"
#include "include/EncryptDecrypt.hxx"
```

Classes

- class [ResourceData](#)
Internal class for resource with more information.

Typedefs

- typedef std::tuple< int, int > [inFilePosition](#)
Define a datatype of its own for the file position in the input file.

9.13.1 Detailed Description

This file contains the class definition of the [ResourceData](#) class. [ResourceData](#) is derived from [Resource](#), but contains a lot more information

Author

Andreas Tscharner

Date

2012-01-07

Definition in file [ResourceData.hxx](#).

9.13.2 Typedef Documentation

9.13.2.1 typedef std::tuple<int, int> inFilePosition

Define a datatype of its own for the file position in the input file.

Definition at line 44 of file [ResourceData.hxx](#).

9.14 /home/andy/Programming/Projects/lrc/src/StatusCodes.hxx File Reference

Macros

- #define [NO_ERROR](#) 0
No error, everything OK.
- #define [WARNING_BASE](#) 0
Base value for all warning codes.
- #define [WARNING_DOUBLE_RESOURCE_ID](#) [WARNING_BASE](#)-1
The resource ID appears more than once.
- #define [ERROR_BASE](#) -1000

- Base value for all error codes.*
- #define [ERROR_FILE_OPEN ERROR_BASE-1](#)
An error occurred while opening the file.
 - #define [ERROR_FILE_NOT_FOUND ERROR_BASE-2](#)
The desired file could not be found.
 - #define [ERROR_FILE_READ ERROR_BASE-3](#)
An error occurred while reading the file.
 - #define [ERROR_FILE_WRITE ERROR_BASE-4](#)
An error occurred while writing the file.
 - #define [ERROR_PARSE ERROR_BASE-10](#)
An error occurred while parsing the file.
 - #define [ERROR_COMPRESSION_NOT_AVAILABLE ERROR_BASE-20](#)
The desired compression is not available.
 - #define [ERROR_ENCRYPTION_NOT_AVAILABLE ERROR_BASE-21](#)
The desired encryption is not available.
 - #define [ERROR_COMPRESSION_COMPRESS ERROR_BASE-22](#)
An error occurred while compressing.
 - #define [ERROR_COMPRESSION_DECOMPRESS ERROR_BASE-23](#)
An error occurred while decompressing.
 - #define [ERROR_ENCRYPTION_ENCRYPT ERROR_BASE-24](#)
An error occurred while encrypting.
 - #define [ERROR_ENCRYPTION_DECRYPT ERROR_BASE-25](#)
An error occurred while decrypting.
 - #define [ERROR_INVALID_PARAMETER ERROR_BASE-100](#)
The provided parameter was illegal.
 - #define [ERROR_NOT_ENOUGH_MEMORY ERROR_BASE-101](#)
Not enough memory available for this action.

Functions

- bool [no_error](#) (int c)
Check for NO_ERROR.
- bool [success](#) (int c)
Check for success.
- bool [is_warning](#) (int c)
Check for warning.
- bool [is_error](#) (int c)
Check for error.

9.14.1 Detailed Description

This file contains a number of defines that are used as status (success, warning or error) codes and a few inline functions for better handling these codes

Author

Andreas Tscharner

Date

2011-09-04

Definition in file [StatusCodes.hxx](#).

9.14.2 Macro Definition Documentation

9.14.2.1 #define ERROR_BASE -1000

Base value for all error codes.

Definition at line 44 of file StatusCodes.hxx.

9.14.2.2 #define ERROR_COMPRESSION_COMPRESS ERROR_BASE-22

An error occurred while compressing.

Definition at line 58 of file StatusCodes.hxx.

9.14.2.3 #define ERROR_COMPRESSION_DECOMPRESS ERROR_BASE-23

An error occurred while decompressing.

Definition at line 59 of file StatusCodes.hxx.

9.14.2.4 #define ERROR_COMPRESSION_NOT_AVAILABLE ERROR_BASE-20

The desired compression is not available.

Definition at line 56 of file StatusCodes.hxx.

9.14.2.5 #define ERROR_ENCRYPTION_DECRYPT ERROR_BASE-25

An error occurred while decrypting.

Definition at line 61 of file StatusCodes.hxx.

9.14.2.6 #define ERROR_ENCRYPTION_ENCRYPT ERROR_BASE-24

An error occurred while encrypting.

Definition at line 60 of file StatusCodes.hxx.

9.14.2.7 #define ERROR_ENCRYPTION_NOT_AVAILABLE ERROR_BASE-21

The desired encryption is not available.

Definition at line 57 of file StatusCodes.hxx.

9.14.2.8 #define ERROR_FILE_NOT_FOUND ERROR_BASE-2

The desired file could not be found.

Definition at line 48 of file StatusCodes.hxx.

9.14.2.9 #define ERROR_FILE_OPEN ERROR_BASE-1

An error occurred while opening the file.

Definition at line 47 of file StatusCodes.hxx.

9.14.2.10 `#define ERROR_FILE_READ ERROR_BASE-3`

An error occurred while reading the file.

Definition at line 49 of file StatusCodes.hxx.

9.14.2.11 `#define ERROR_FILE_WRITE ERROR_BASE-4`

An error occurred while writing the file.

Definition at line 50 of file StatusCodes.hxx.

9.14.2.12 `#define ERROR_INVALID_PARAMETER ERROR_BASE-100`

The provided parameter was illegal.

Definition at line 64 of file StatusCodes.hxx.

9.14.2.13 `#define ERROR_NOT_ENOUGH_MEMORY ERROR_BASE-101`

Not enough memory available for this action.

Definition at line 65 of file StatusCodes.hxx.

9.14.2.14 `#define ERROR_PARSE ERROR_BASE-10`

An error occurred while parsing the file.

Definition at line 53 of file StatusCodes.hxx.

9.14.2.15 `#define NO_ERROR 0`

No error, everything OK.

Definition at line 37 of file StatusCodes.hxx.

9.14.2.16 `#define WARNING_BASE 0`

Base value for all warning codes.

Definition at line 40 of file StatusCodes.hxx.

9.14.2.17 `#define WARNING_DOUBLE_RESOURCE_ID WARNING_BASE-1`

The resource ID appears more than once.

Definition at line 41 of file StatusCodes.hxx.

9.14.3 Function Documentation

9.14.3.1 `bool is_error (int c) [inline]`

Check for error.

Small inline function to check if the given parameter stands for an error

Parameters

<i>in</i>	<i>c</i>	Value to check
-----------	----------	----------------

Return values

<i>true</i>	Value is an error code
<i>false</i>	Value stands for warning or success

Definition at line 112 of file StatusCodes.hxx.

9.14.3.2 bool is_warning (int c) [inline]

Check for warning.

Small inline function to check if the given parameter stands for a warning

Parameters

<i>in</i>	<i>c</i>	Value to check
-----------	----------	----------------

Return values

<i>true</i>	Value is a warning code
<i>false</i>	Value stands for success or error

Definition at line 101 of file StatusCodes.hxx.

9.14.3.3 bool no_error (int c) [inline]

Check for NO_ERROR.

Small inline function to check if the given parameter is NO_ERROR

Parameters

<i>in</i>	<i>c</i>	Value to check
-----------	----------	----------------

Return values

<i>true</i>	Value is NO_ERROR
<i>false</i>	Value is not NO_ERROR

Definition at line 78 of file StatusCodes.hxx.

9.14.3.4 bool success (int c) [inline]

Check for success.

Small inline function to check if the given parameter means success, e.g. is greater than NO_ERROR

Parameters

<i>in</i>	<i>c</i>	Value to check
-----------	----------	----------------

Return values

<i>true</i>	Value means success
<i>false</i>	Value means warning or error

Definition at line 89 of file StatusCodes.hxx.

9.15 /home/andy/Programming/Projects/lrc/src/strategies/bz2LibCompression.hxx File Reference

```
#include "../include/CompressDecompress.hxx"
```

Classes

- class [bz2LibCompression](#)
Compression class that uses the bzip2 algorithm.

9.15.1 Detailed Description

This file contains the declaration of the bz2Lib compression class. This class is used to compress and decompress the resource data using the bzip2 algorithm.

Author

Andreas Tscharner

Date

2012-01-18

Definition in file [bz2LibCompression.hxx](#).

9.16 /home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.hxx File Reference

```
#include "../include/CompressDecompress.hxx"
```

Classes

- class [NoneCompression](#)
Compression class that does NO compression.

9.16.1 Detailed Description

This file contains the class declaration of the [NoneCompression](#) class. This class is a dummy class that does no compression at all, but is used to fit in the Strategy Pattern

Author

Andreas Tscharner

Date

2011-06-25

Definition in file [NoneCompression.hxx](#).

9.17 /home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx File Reference

```
#include "../include/EncryptDecrypt.hxx"
```

Classes

- class [NoneEncryption](#)
Encryption class that does NO encryption.

9.17.1 Detailed Description

This file contains the [NoneEncryption](#) class. This class does no encryption at all, but is used to fit in the Strategy Pattern

Author

Andreas Tscharner

Date

0211-07-21

Definition in file [NoneEncryption.hxx](#).

9.18 /home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx File Reference

```
#include "../include/EncryptDecrypt.hxx"
```

Classes

- class [SerpentEncryption](#)
Class to encrypt/decrypt using the Serpent algorithm.

9.18.1 Detailed Description

This file contains the declaration of the `SerpenEncryption` class. This class is used to encrypt and decrypt the data using the *Serpent* algorithm

Author

Andreas Tschärner

Date

2011-08-24

Definition in file [SerpentEncryption.hxx](#).

9.19 /home/andy/Programming/Projects/lrc/src/strategies/zLibCompression.hxx File Reference

```
#include "../include/CompressDecompress.hxx"
```

Classes

- class [zLibCompression](#)
Compression class that uses the zLib algorithm.

9.19.1 Detailed Description

This file contains the declaration of the `zLib` compression class. This class is used to compress and decompress the resource data using the `zLib` algorithm.

Author

Andreas Tschärner

Date

2011-08-14

Definition in file [zLibCompression.hxx](#).

9.20 /home/andy/Programming/Projects/lrc/src/Utils.hxx File Reference

Macros

- `#define DEBUG_PRINT\(x\)`
Macro for debugging.

Functions

- bool [file_exists](#) (const char *)
Check if a file exists.
- int [file_size](#) (const char *)
Returns the size of a file.
- char * [get_extension](#) (char *)
Returns the file extension.
- char * [replace_extension](#) (char *, char *)
Replace the file extension.
- void [delete_list](#) (unsigned char **, unsigned int)
Delete list of data.
- int [password_len](#) (const unsigned char *)
Gets length of a password.

9.20.1 Detailed Description

This file contains the declaration of a few utility functions that are used in the whole project

Author

Andreas Tscharner

Date

2011-09-11

Definition in file [Utils.hxx](#).

9.20.2 Macro Definition Documentation

9.20.2.1 #define DEBUG_PRINT(x)

Macro for debugging.

This macro expands to a printf if the **DEBUG** define is set or to nothing otherwise

Definition at line 43 of file [Utils.hxx](#).

9.20.3 Function Documentation

9.20.3.1 void delete_list (unsigned char **, unsigned int)

Delete list of data.

This function deletes a list of data (double pointers to unsigned chars)

Parameters

in	<i>p_dataList</i>	List of data to delete
in	<i>p_listSize</i>	Size of data list

9.20.3.2 bool file_exists (const char *)

Check if a file exists.

This function checks if a given file exists and returns true if it does and false otherwise

Parameters

<i>in</i>	<i>p_filename</i>	Full or relative path to file
-----------	-------------------	-------------------------------

Return values

<i>true</i>	File exists
<i>false</i>	File does not exist

9.20.3.3 int file_size (const char *)

Returns the size of a file.

This function returns the size of a file

Parameters

<i>in</i>	<i>p_filename</i>	Filename
-----------	-------------------	----------

Returns

Size of file

Remarks

If an error occurs, the function returns -1

9.20.3.4 char* get_extension (char *)

Returns the file extension.

This function returns the file extension of the given file

Parameters

<i>in</i>	<i>p_filename</i>	Filename
-----------	-------------------	----------

Returns

File extension (if any)

Remarks

The returned string is a pointer to the file extension within the given string! It is NULL, if the given filename has no extension.

9.20.3.5 int password_len (const unsigned char *)

Gets length of a password.

This function is used to get the length of a zero-terminated password. It counts all *unsigned* characters until the first zero

Parameters

<code>in</code>	<code>p_password</code>	Zero-terminated password
-----------------	-------------------------	--------------------------

Returns

Length of zero-terminated password

Remarks

-1 will be returned if the given password was `nullptr`

9.20.3.6 char* replace_extension (char *, char *)

Replace the file extension.

This function replaces the extension of the given file with the given new extension. The extension is expected to have a dot ('.') as first character. If the given filename has no extension, the new extension is appended.

Parameters

<code>in</code>	<code>p_filename</code>	Filename
<code>in</code>	<code>p_newExt</code>	New file extension

Returns

Filename with new extension

Remarks

The caller is responsible to free the allocated memory of the returned string

Index

- ~Collector
 - Collector, 17
- ~InFileParser
 - InFileParser, 25
- ~Resource
 - Irc::Resource, 39
- ~ResourceData
 - ResourceData, 41
- ~ResourceManager
 - Irc::ResourceManager, 47
- ~IrcEncryptionDisabledException
 - IrcEncryptionDisabledException, 29
- ~IrcFileExistsException
 - IrcFileExistsException, 30
- ~IrcFileNotFoundException
 - IrcFileNotFoundException, 32
- /home/andy/Programming/Projects/lrc/src/Factoryes.-
hxx, 58
- /home/andy/Programming/Projects/lrc/src/Resource-
Data.hxx, 63
- /home/andy/Programming/Projects/lrc/src/Status-
Codes.hxx, 63
- /home/andy/Programming/Projects/lrc/src/Utils.hxx, 70
- /home/andy/Programming/Projects/lrc/src/compiler/-
Collector.hxx, 55
- /home/andy/Programming/Projects/lrc/src/compiler/In-
FileParser.hxx, 55
- /home/andy/Programming/Projects/lrc/src/compiler/-
ParserFactory.hxx, 57
- /home/andy/Programming/Projects/lrc/src/compiler/RC-
Parser.hxx, 57
- /home/andy/Programming/Projects/lrc/src/compiler/RIF-
Parser.hxx, 58
- /home/andy/Programming/Projects/lrc/src/compiler/lrc.-
cxx, 56
- /home/andy/Programming/Projects/lrc/src/include/-
CompressDecompress.hxx, 59
- /home/andy/Programming/Projects/lrc/src/include/-
EncryptDecrypt.hxx, 60
- /home/andy/Programming/Projects/lrc/src/include/-
Resource.hxx, 60
- /home/andy/Programming/Projects/lrc/src/include/-
ResourceManager.hxx, 61
- /home/andy/Programming/Projects/lrc/src/lrcExceptions.-
hxx, 62
- /home/andy/Programming/Projects/lrc/src/strategies/-
NoneCompression.hxx, 68
- /home/andy/Programming/Projects/lrc/src/strategies/-
NoneEncryption.hxx, 69
- /home/andy/Programming/Projects/lrc/src/strategies/-
SerpentEncryption.hxx, 69
- /home/andy/Programming/Projects/lrc/src/strategies/bz2-
LibCompression.hxx, 68
- /home/andy/Programming/Projects/lrc/src/strategies/z-
LibCompression.hxx, 70
- are_resIDs_unique
 - Collector, 17
- bz2LibCompression
 - Irc, 14
- bz2LibCompression, 15
 - compress, 15
 - decompress, 15
- clear_internal_error
 - InFileParser, 25
- collect
 - Collector, 18
- Collector, 16
 - ~Collector, 17
 - are_resIDs_unique, 17
 - collect, 18
 - Collector, 17
 - m_rcName, 18
 - m_rdfName, 19
 - show_resource_data_error, 18
- compType
 - resEntry_, 37
- compress
 - bz2LibCompression, 15
 - Irc::CompressDecompress, 19
 - NoneCompression, 33
 - zLibCompression, 52
- CompressionFactory, 20
 - get_compression_class, 21
- CompressionType
 - Irc, 13
- create_initialization_vector
 - SerpentEncryption, 51
- create_input_parser
 - ParserFactory, 34
- DEBUG_PRINT
 - Utils.hxx, 71
- decompress
 - bz2LibCompression, 15
 - Irc::CompressDecompress, 20
 - NoneCompression, 33

- zLibCompression, 52
- decrypt
 - Irc::EncryptDecrypt, 21
 - NoneEncryption, 33
 - SerpentEncryption, 51
- delete_list
 - Utils.hxx, 71
- ERROR_BASE
 - StatusCodes.hxx, 65
- ERROR_FILE_OPEN
 - StatusCodes.hxx, 65
- ERROR_FILE_READ
 - StatusCodes.hxx, 65
- ERROR_FILE_WRITE
 - StatusCodes.hxx, 66
- ERROR_PARSE
 - StatusCodes.hxx, 66
- encType
 - resEntry_, 37
- encrypt
 - Irc::EncryptDecrypt, 22
 - NoneEncryption, 34
 - SerpentEncryption, 51
- EncryptionFactory, 22
 - get_encryption_class, 23
- EncryptionType
 - Irc, 14
- eval_compression_type
 - InFileParser, 25
- eval_encryption_type
 - InFileParser, 26
- file_exists
 - Utils.hxx, 71
- file_size
 - Utils.hxx, 72
- get_ID
 - Irc::Resource, 39
- get_compression
 - ResourceData, 41
- get_compression_class
 - CompressionFactory, 21
- get_data_from_memory
 - ResourceData, 42
- get_encryption
 - ResourceData, 42
- get_encryption_class
 - EncryptionFactory, 23
- get_error_msg
 - ResourceData, 42
- get_extension
 - Utils.hxx, 72
- get_file
 - ResourceData, 43
- get_internal_error
 - InFileParser, 26
- get_password
 - InFileParser, 27
 - ResourceData, 43
- get_res_data
 - Irc::Resource, 39
- get_res_size
 - Irc::Resource, 39
- get_resource
 - Irc::ResourceManager, 47
- get_resource_entries
 - InFileParser, 27
- get_resource_ids
 - Irc::ResourceManager, 48
- ieFilenameNotFound
 - InFileParser, 25
- ieIdentNotFound
 - InFileParser, 25
- ieInvalidElement
 - InFileParser, 25
- ieMissingPassword
 - InFileParser, 25
- ieNone
 - InFileParser, 25
- iePasswordFileNotFound
 - InFileParser, 25
- InFileParser
 - ieFilenameNotFound, 25
 - ieIdentNotFound, 25
 - ieInvalidElement, 25
 - ieMissingPassword, 25
 - ieNone, 25
 - iePasswordFileNotFound, 25
- InFileParser, 23
 - ~InFileParser, 25
 - clear_internal_error, 25
 - eval_compression_type, 26
 - eval_encryption_type, 26
 - get_internal_error, 26
 - get_password, 27
 - get_resource_entries, 27
 - InFileParser, 25
 - InFileParser, 25
 - internalErrorType, 25
 - m_errorPosition, 28
 - m_filename, 28
 - m_internalError, 28
 - m_lastError, 28
 - m_resEntries, 28
 - parse, 27
- inFilePosition
 - ResourceData.hxx, 63
- internalErrorType
 - InFileParser, 25
- is_comment
 - RCParse, 36
- is_error
 - StatusCodes.hxx, 66
- is_warning

- StatusCodes.hxx, 67
- load_from_file
 - Irc::ResourceManager, 48
- Irc, 13
 - bz2LibCompression, 14
 - CompressionType, 13
 - EncryptionType, 14
 - NoneCompression, 14
 - NoneEncryption, 14
 - SerpentEncryption, 14
 - zLibCompression, 14
- Irc.cxx
 - main, 57
 - usage, 57
 - VERSION, 57
- Irc::CompressDecompress, 19
 - compress, 19
 - decompress, 20
- Irc::EncryptDecrypt, 21
 - decrypt, 21
 - encrypt, 22
- Irc::Resource, 38
 - ~Resource, 39
 - get_ID, 39
 - get_res_data, 39
 - get_res_size, 39
 - m_resData, 39
 - m_resID, 39
 - m_resSize, 39
 - Resource, 38
- Irc::ResourceManager, 46
 - ~ResourceManager, 47
 - get_resource, 47
 - get_resource_ids, 48
 - load_from_file, 48
 - m_compType, 48
 - m_encType, 48
 - m_numResEntries, 48
 - m_password, 48
 - m_resData, 49
 - m_resDataSize, 49
 - m_resEntries, 49
 - m_resourceFile, 49
 - ResourceManager, 47
- IrcEncryptionDisabledException, 28
 - ~IrcEncryptionDisabledException, 29
 - IrcEncryptionDisabledException, 29
 - IrcEncryptionDisabledException, 29
 - m_resourceID, 29
 - what, 29
- IrcFileExistsException, 30
 - ~IrcFileExistsException, 30
 - IrcFileExistsException, 30
 - IrcFileExistsException, 30
 - m_fileOverwrite, 31
 - what, 30
- IrcFileNotFoundException, 31
 - ~IrcFileNotFoundException, 32
- IrcFileNotFoundException, 31
 - IrcFileNotFoundException, 31
 - m_fileNotFound, 32
 - what, 32
- m_compType
 - Irc::ResourceManager, 48
- m_compression
 - ResourceData, 45
- m_encType
 - Irc::ResourceManager, 48
- m_encryption
 - ResourceData, 45
- m_errorMsg
 - ResourceData, 45
- m_errorPosition
 - InFileParser, 28
- m_fileNotFound
 - IrcFileNotFoundException, 32
- m_fileOverwrite
 - IrcFileExistsException, 31
- m_filename
 - InFileParser, 28
 - ResourceData, 45
- m_inFilePosition
 - ResourceData, 45
- m_internalError
 - InFileParser, 28
- m_lastError
 - InFileParser, 28
- m_numResEntries
 - Irc::ResourceManager, 48
- m_password
 - Irc::ResourceManager, 48
 - ResourceData, 45
- m_rcName
 - Collector, 18
- m_rdfName
 - Collector, 19
- m_resData
 - Irc::Resource, 39
 - Irc::ResourceManager, 49
- m_resDataSize
 - Irc::ResourceManager, 49
- m_resEntries
 - InFileParser, 28
 - Irc::ResourceManager, 49
- m_resID
 - Irc::Resource, 39
- m_resSize
 - Irc::Resource, 39
- m_resourceFile
 - Irc::ResourceManager, 49
- m_resourceID
 - IrcEncryptionDisabledException, 29
- MAX_ID_LEN
 - Resource.hxx, 61
- main
 - Irc.cxx, 57

- NO_ERROR
 - StatusCodes.hxx, 66
- no_error
 - StatusCodes.hxx, 67
- NoneCompression
 - Irc, 14
- NoneEncryption
 - Irc, 14
- NoneCompression, 32
 - compress, 33
 - decompress, 33
- NoneEncryption, 33
 - decrypt, 33
 - encrypt, 34
- parse
 - InFileParser, 27
 - RCParse, 36
 - RIFParser, 50
- ParserFactory, 34
 - create_input_parser, 34
- password_len
 - Utils.hxx, 72
- prepare_resource_from_file
 - ResourceData, 43
- RCParse, 35
 - is_comment, 36
 - parse, 36
 - RCParse, 35
 - RCParse, 35
- RIFParser, 49
 - parse, 50
 - RIFParser, 50
 - RIFParser, 50
- replace_extension
 - Utils.hxx, 73
- resEntry
 - Resource.hxx, 61
- resEntry_, 36
 - compType, 37
 - encType, 37
 - resID, 37
 - resSize, 37
 - startOffset, 37
- resID
 - resEntry_, 37
- resSize
 - resEntry_, 37
- Resource
 - Irc::Resource, 38
- Resource.hxx
 - MAX_ID_LEN, 61
 - resEntry, 61
- ResourceData, 40
 - ~ResourceData, 41
 - get_compression, 41
 - get_data_from_memory, 42
 - get_encryption, 42
 - get_error_msg, 42
 - get_file, 43
 - get_rc_position, 43
 - m_compression, 45
 - m_encryption, 45
 - m_errorMsg, 45
 - m_filename, 45
 - m_inFilePosition, 45
 - m_password, 45
 - prepare_resource_from_file, 43
 - ResourceData, 41
 - ResourceData, 41
 - set_compression, 43
 - set_encryption, 44
 - set_error_msg, 44
 - set_file, 44
 - set_ident, 44
 - set_rc_position, 44
- ResourceData.hxx
 - inFilePosition, 63
- ResourceManager
 - Irc::ResourceManager, 47
- SerpentEncryption
 - Irc, 14
- SerpentEncryption, 50
 - create_initialization_vector, 51
 - decrypt, 51
 - encrypt, 51
- set_compression
 - ResourceData, 43
- set_encryption
 - ResourceData, 44
- set_error_msg
 - ResourceData, 44
- set_file
 - ResourceData, 44
- set_ident
 - ResourceData, 44
- set_rc_position
 - ResourceData, 44
- show_resource_data_error
 - Collector, 18
- startOffset
 - resEntry_, 37
- StatusCodes.hxx
 - ERROR_BASE, 65
 - ERROR_FILE_OPEN, 65
 - ERROR_FILE_READ, 65
 - ERROR_FILE_WRITE, 66
 - ERROR_PARSE, 66
 - is_error, 66
 - is_warning, 67
 - NO_ERROR, 66
 - no_error, 67
 - success, 67
 - WARNING_BASE, 66
- success
 - StatusCodes.hxx, 67

usage

- lrc.cxx, [57](#)

Utils.hxx

- DEBUG_PRINT, [71](#)
- delete_list, [71](#)
- file_exists, [71](#)
- file_size, [72](#)
- get_extension, [72](#)
- password_len, [72](#)
- replace_extension, [73](#)

VERSION

- lrc.cxx, [57](#)

WARNING_BASE

- StatusCodes.hxx, [66](#)

what

- IrcEncryptionDisabledException, [29](#)
- IrcFileExistsException, [30](#)
- IrcFileNotFoundException, [32](#)

zLibCompression

- lrc, [14](#)

zLibCompression, [52](#)

- compress, [52](#)
- decompress, [52](#)