

# Irc - Linux Resource Compiler

1.2.1

Generated by Doxygen 1.8.13



# 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 . . . . . 2
  
- 2 Todo List** **3**
  
- 3 Namespace Index** **5**
  - 3.1 Namespace List . . . . . 5
  
- 4 Hierarchical 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 . . . . . 14
      - 7.1.2.1 CompressionType . . . . . 14
      - 7.1.2.2 EncryptionType . . . . . 14

<b>8 Class Documentation</b>	<b>15</b>
8.1 bz2LibCompression Class Reference	15
8.1.1 Detailed Description	16
8.1.2 Member Function Documentation	16
8.1.2.1 compress()	16
8.1.2.2 decompress()	16
8.2 Collector Class Reference	17
8.2.1 Detailed Description	17
8.2.2 Constructor & Destructor Documentation	17
8.2.2.1 Collector()	17
8.2.2.2 ~Collector()	18
8.2.3 Member Function Documentation	18
8.2.3.1 are_resIDs_unique()	18
8.2.3.2 collect()	19
8.2.3.3 show_resource_data_error()	20
8.2.4 Member Data Documentation	20
8.2.4.1 m_rcName	20
8.2.4.2 m_rdfName	20
8.3 Irc::CompressDecompress Class Reference	21
8.3.1 Detailed Description	21
8.3.2 Member Function Documentation	21
8.3.2.1 compress()	21
8.3.2.2 decompress()	22
8.4 CompressionFactory Class Reference	23
8.4.1 Detailed Description	23
8.4.2 Member Function Documentation	23
8.4.2.1 get_compression_class()	23
8.5 Irc::EncryptDecrypt Class Reference	24
8.5.1 Detailed Description	24
8.5.2 Member Function Documentation	24

---

8.5.2.1	decrypt()	24
8.5.2.2	encrypt()	25
8.6	EncryptionFactory Class Reference	26
8.6.1	Detailed Description	26
8.6.2	Member Function Documentation	26
8.6.2.1	get_encryption_class()	26
8.7	InFileParser Class Reference	27
8.7.1	Detailed Description	29
8.7.2	Member Enumeration Documentation	29
8.7.2.1	internalErrorType	29
8.7.3	Constructor & Destructor Documentation	30
8.7.3.1	InFileParser()	30
8.7.3.2	~InFileParser()	30
8.7.4	Member Function Documentation	31
8.7.4.1	clear_internal_error()	31
8.7.4.2	eval_compression_type()	31
8.7.4.3	eval_encryption_type()	31
8.7.4.4	get_internal_error()	32
8.7.4.5	get_password()	32
8.7.4.6	get_resource_entries()	33
8.7.4.7	parse()	33
8.7.5	Member Data Documentation	34
8.7.5.1	m_errorPosition	34
8.7.5.2	m_filename	34
8.7.5.3	m_internalError	34
8.7.5.4	m_lastError	34
8.7.5.5	m_resEntries	35
8.8	IrcEncryptionDisabledException Class Reference	35
8.8.1	Detailed Description	36
8.8.2	Constructor & Destructor Documentation	36

---

---

8.8.2.1	<a href="#">IrcEncryptionDisabledException()</a> . . . . .	36
8.8.2.2	<a href="#">~IrcEncryptionDisabledException()</a> . . . . .	36
8.8.3	<a href="#">Member Function Documentation</a> . . . . .	37
8.8.3.1	<a href="#">what()</a> . . . . .	37
8.8.4	<a href="#">Member Data Documentation</a> . . . . .	37
8.8.4.1	<a href="#">m_resourceID</a> . . . . .	37
8.9	<a href="#">IrcFileExistsException Class Reference</a> . . . . .	38
8.9.1	<a href="#">Detailed Description</a> . . . . .	39
8.9.2	<a href="#">Constructor &amp; Destructor Documentation</a> . . . . .	39
8.9.2.1	<a href="#">IrcFileExistsException()</a> . . . . .	39
8.9.2.2	<a href="#">~IrcFileExistsException()</a> . . . . .	39
8.9.3	<a href="#">Member Function Documentation</a> . . . . .	39
8.9.3.1	<a href="#">what()</a> . . . . .	39
8.9.4	<a href="#">Member Data Documentation</a> . . . . .	40
8.9.4.1	<a href="#">m_fileOverwrite</a> . . . . .	40
8.10	<a href="#">IrcFileNotFoundException Class Reference</a> . . . . .	40
8.10.1	<a href="#">Detailed Description</a> . . . . .	41
8.10.2	<a href="#">Constructor &amp; Destructor Documentation</a> . . . . .	41
8.10.2.1	<a href="#">IrcFileNotFoundException()</a> . . . . .	41
8.10.2.2	<a href="#">~IrcFileNotFoundException()</a> . . . . .	41
8.10.3	<a href="#">Member Function Documentation</a> . . . . .	42
8.10.3.1	<a href="#">what()</a> . . . . .	42
8.10.4	<a href="#">Member Data Documentation</a> . . . . .	42
8.10.4.1	<a href="#">m_fileNotFound</a> . . . . .	42
8.11	<a href="#">NoneCompression Class Reference</a> . . . . .	42
8.11.1	<a href="#">Detailed Description</a> . . . . .	43
8.11.2	<a href="#">Member Function Documentation</a> . . . . .	43
8.11.2.1	<a href="#">compress()</a> . . . . .	43
8.11.2.2	<a href="#">decompress()</a> . . . . .	44
8.12	<a href="#">NoneEncryption Class Reference</a> . . . . .	44

---

---

8.12.1 Detailed Description . . . . .	45
8.12.2 Member Function Documentation . . . . .	45
8.12.2.1 decrypt() . . . . .	45
8.12.2.2 encrypt() . . . . .	45
8.13 ParserFactory Class Reference . . . . .	46
8.13.1 Detailed Description . . . . .	46
8.13.2 Member Function Documentation . . . . .	46
8.13.2.1 create_input_parser() . . . . .	46
8.14 RCParse Class Reference . . . . .	47
8.14.1 Detailed Description . . . . .	48
8.14.2 Constructor & Destructor Documentation . . . . .	48
8.14.2.1 RCParse() . . . . .	48
8.14.3 Member Function Documentation . . . . .	49
8.14.3.1 copy_mandatory_data() . . . . .	49
8.14.3.2 is_comment() . . . . .	49
8.14.3.3 is_windows_line() . . . . .	50
8.14.3.4 parse() . . . . .	50
8.15 resEntry_ Struct Reference . . . . .	51
8.15.1 Detailed Description . . . . .	51
8.15.2 Member Data Documentation . . . . .	51
8.15.2.1 compType . . . . .	51
8.15.2.2 encType . . . . .	51
8.15.2.3 resID . . . . .	52
8.15.2.4 resSize . . . . .	52
8.15.2.5 startOffset . . . . .	52
8.16 Irc::Resource Class Reference . . . . .	52
8.16.1 Detailed Description . . . . .	53
8.16.2 Constructor & Destructor Documentation . . . . .	53
8.16.2.1 Resource() . . . . .	53
8.16.2.2 ~Resource() . . . . .	54

8.16.3	Member Function Documentation	54
8.16.3.1	get_ID()	54
8.16.3.2	get_res_data()	54
8.16.3.3	get_res_size()	54
8.16.4	Member Data Documentation	55
8.16.4.1	m_resData	55
8.16.4.2	m_resID	55
8.16.4.3	m_resSize	55
8.17	ResourceData Class Reference	56
8.17.1	Detailed Description	57
8.17.2	Constructor & Destructor Documentation	58
8.17.2.1	ResourceData()	58
8.17.2.2	~ResourceData()	58
8.17.3	Member Function Documentation	58
8.17.3.1	get_compression()	58
8.17.3.2	get_data_from_memory()	58
8.17.3.3	get_encryption()	59
8.17.3.4	get_error_msg()	59
8.17.3.5	get_file()	60
8.17.3.6	get_rc_position()	60
8.17.3.7	prepare_resource_from_file()	60
8.17.3.8	set_compression()	61
8.17.3.9	set_encryption()	61
8.17.3.10	set_error_msg()	62
8.17.3.11	set_file()	62
8.17.3.12	set_ident()	62
8.17.3.13	set_rc_position()	62
8.17.4	Member Data Documentation	63
8.17.4.1	m_compression	63
8.17.4.2	m_encryption	63



---

8.17.4.3	<code>m_errorMsg</code>	63
8.17.4.4	<code>m_filename</code>	64
8.17.4.5	<code>m_inFilePosition</code>	64
8.17.4.6	<code>m_password</code>	64
8.18	<code>Irc::ResourceManager</code> Class Reference	64
8.18.1	Detailed Description	66
8.18.2	Constructor & Destructor Documentation	66
8.18.2.1	<code>ResourceManager()</code> [1/2]	66
8.18.2.2	<code>ResourceManager()</code> [2/2]	66
8.18.2.3	<code>~ResourceManager()</code>	67
8.18.3	Member Function Documentation	67
8.18.3.1	<code>decompress_data()</code>	67
8.18.3.2	<code>decrypt_data()</code>	68
8.18.3.3	<code>get_resource()</code> [1/2]	68
8.18.3.4	<code>get_resource()</code> [2/2]	69
8.18.3.5	<code>get_resource_ids()</code>	69
8.18.3.6	<code>load_embedded()</code>	70
8.18.3.7	<code>load_from_file()</code>	70
8.18.3.8	<code>setup_resources()</code>	71
8.18.4	Member Data Documentation	71
8.18.4.1	<code>m_compType</code>	71
8.18.4.2	<code>m_encType</code>	71
8.18.4.3	<code>m_numResEntries</code>	71
8.18.4.4	<code>m_password</code>	72
8.18.4.5	<code>m_resData</code>	72
8.18.4.6	<code>m_resDataSize</code>	72
8.18.4.7	<code>m_resEntries</code>	72
8.18.4.8	<code>m_resourceFile</code>	72
8.19	<code>RIFParser</code> Class Reference	73
8.19.1	Detailed Description	74

---

8.19.2	Constructor & Destructor Documentation	74
8.19.2.1	RIFParser()	74
8.19.3	Member Function Documentation	74
8.19.3.1	parse()	74
8.20	SerpentEncryption Class Reference	75
8.20.1	Detailed Description	76
8.20.2	Member Function Documentation	76
8.20.2.1	create_initialization_vector()	76
8.20.2.2	decrypt()	77
8.20.2.3	encrypt()	77
8.21	zLibCompression Class Reference	78
8.21.1	Detailed Description	78
8.21.2	Member Function Documentation	79
8.21.2.1	compress()	79
8.21.2.2	decompress()	79
<b>9</b>	<b>File Documentation</b>	<b>81</b>
9.1	/home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx File Reference	81
9.1.1	Detailed Description	82
9.2	/home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx File Reference	82
9.2.1	Detailed Description	83
9.3	/home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx File Reference	84
9.3.1	Detailed Description	85
9.3.2	Macro Definition Documentation	85
9.3.2.1	VERSION	85
9.3.3	Function Documentation	85
9.3.3.1	convert_to_elf()	85
9.3.3.2	main()	85
9.3.3.3	usage()	86
9.4	/home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx File Reference	86
9.4.1	Detailed Description	87

---

9.5	<a href="#">/home/andy/Programming/Projects/lrc/src/compiler/RCParse.hxx File Reference</a>	87
9.5.1	<a href="#">Detailed Description</a>	88
9.6	<a href="#">/home/andy/Programming/Projects/lrc/src/compiler/RIFParse.hxx File Reference</a>	89
9.6.1	<a href="#">Detailed Description</a>	89
9.7	<a href="#">/home/andy/Programming/Projects/lrc/src/Factory.hxx File Reference</a>	90
9.7.1	<a href="#">Detailed Description</a>	90
9.8	<a href="#">/home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx File Reference</a>	91
9.8.1	<a href="#">Detailed Description</a>	92
9.9	<a href="#">/home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx File Reference</a>	92
9.9.1	<a href="#">Detailed Description</a>	93
9.10	<a href="#">/home/andy/Programming/Projects/lrc/src/include/Resource.hxx File Reference</a>	93
9.10.1	<a href="#">Detailed Description</a>	95
9.10.2	<a href="#">Macro Definition Documentation</a>	95
9.10.2.1	<a href="#">MAX_ID_LEN</a>	95
9.10.3	<a href="#">Typedef Documentation</a>	95
9.10.3.1	<a href="#">resEntry</a>	95
9.11	<a href="#">/home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx File Reference</a>	96
9.11.1	<a href="#">Detailed Description</a>	96
9.12	<a href="#">/home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx File Reference</a>	97
9.12.1	<a href="#">Detailed Description</a>	97
9.13	<a href="#">/home/andy/Programming/Projects/lrc/src/ResourceData.hxx File Reference</a>	98
9.13.1	<a href="#">Detailed Description</a>	99
9.13.2	<a href="#">Typedef Documentation</a>	99
9.13.2.1	<a href="#">inFilePosition</a>	99
9.14	<a href="#">/home/andy/Programming/Projects/lrc/src/StatusCodes.hxx File Reference</a>	99
9.14.1	<a href="#">Detailed Description</a>	101
9.14.2	<a href="#">Macro Definition Documentation</a>	101
9.14.2.1	<a href="#">ERROR_BASE</a>	101
9.14.2.2	<a href="#">ERROR_COMPRESSION_COMPRESS</a>	101
9.14.2.3	<a href="#">ERROR_COMPRESSION_DECOMPRESS</a>	101

9.14.2.4	<a href="#">ERROR_COMPRESSION_NOT_AVAILABLE</a>	102
9.14.2.5	<a href="#">ERROR_ENCRYPTION_DECRYPT</a>	102
9.14.2.6	<a href="#">ERROR_ENCRYPTION_ENCRYPT</a>	102
9.14.2.7	<a href="#">ERROR_ENCRYPTION_NOT_AVAILABLE</a>	102
9.14.2.8	<a href="#">ERROR_FILE_NOT_FOUND</a>	102
9.14.2.9	<a href="#">ERROR_FILE_OPEN</a>	103
9.14.2.10	<a href="#">ERROR_FILE_READ</a>	103
9.14.2.11	<a href="#">ERROR_FILE_WRITE</a>	103
9.14.2.12	<a href="#">ERROR_INVALID_PARAMETER</a>	103
9.14.2.13	<a href="#">ERROR_NOT_ENOUGH_MEMORY</a>	103
9.14.2.14	<a href="#">ERROR_PARSE</a>	104
9.14.2.15	<a href="#">ERROR_UNKNOWN_ARCH</a>	104
9.14.2.16	<a href="#">NO_ERROR</a>	104
9.14.2.17	<a href="#">WARNING_BASE</a>	104
9.14.2.18	<a href="#">WARNING_DOUBLE_RESOURCE_ID</a>	104
9.14.3	<a href="#">Function Documentation</a>	104
9.14.3.1	<a href="#">is_error()</a>	104
9.14.3.2	<a href="#">is_warning()</a>	105
9.14.3.3	<a href="#">no_error()</a>	105
9.14.3.4	<a href="#">success()</a>	106
9.15	<a href="#">/home/andy/Programming/Projects/lrc/src/strategies/bz2LibCompression.hxx File Reference</a>	106
9.15.1	<a href="#">Detailed Description</a>	107
9.16	<a href="#">/home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.hxx File Reference</a>	107
9.16.1	<a href="#">Detailed Description</a>	108
9.17	<a href="#">/home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx File Reference</a>	108
9.17.1	<a href="#">Detailed Description</a>	109
9.18	<a href="#">/home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx File Reference</a>	109
9.18.1	<a href="#">Detailed Description</a>	110
9.19	<a href="#">/home/andy/Programming/Projects/lrc/src/strategies/zLibCompression.hxx File Reference</a>	110
9.19.1	<a href="#">Detailed Description</a>	111
9.20	<a href="#">/home/andy/Programming/Projects/lrc/src/Utils.hxx File Reference</a>	112
9.20.1	<a href="#">Detailed Description</a>	113
9.20.2	<a href="#">Macro Definition Documentation</a>	113
9.20.2.1	<a href="#">DEBUG_PRINT</a>	113
9.20.3	<a href="#">Function Documentation</a>	113
9.20.3.1	<a href="#">delete_list()</a>	113
9.20.3.2	<a href="#">file_exists()</a>	114
9.20.3.3	<a href="#">file_size()</a>	114
9.20.3.4	<a href="#">get_extension()</a>	114
9.20.3.5	<a href="#">password_len()</a>	115
9.20.3.6	<a href="#">replace_extension()</a>	115

# 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] [-m] [-o <rdfFile>] [-c <compression type>]<RCFile>|<resourceFile>
```

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.

`-m`: Prepare the data for direct linking. During the compilation another file with the extension `.o` is created. This file can be linked directly to the executable. The labels `_binary_file_start` and `_binary_file_end` are pointers for the start and the end within the executable. See `ShowEmbeddedImage.cxx` for more information.

`-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**

2014-08-22

## 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:

<a href="#">lrc</a>	Namespace lrc for classes in the library and for extending <code>lrc</code> . . . . .	<a href="#">13</a>
---------------------	---	--------------------



# Chapter 4

## Hierarchical Index

### 4.1 Class Hierarchy

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

Collector . . . . .	17
Irc::CompressDecompress . . . . .	21
bz2LibCompression . . . . .	15
NoneCompression . . . . .	42
zLibCompression . . . . .	78
CompressionFactory . . . . .	23
Irc::EncryptDecrypt . . . . .	24
NoneEncryption . . . . .	44
SerpentEncryption . . . . .	75
EncryptionFactory . . . . .	26
std::exception	
IrcEncryptionDisabledException . . . . .	35
IrcFileExistsException . . . . .	38
IrcFileNotFoundException . . . . .	40
InFileParser . . . . .	27
RCParser . . . . .	47
RIFParser . . . . .	73
ParserFactory . . . . .	46
resEntry_ . . . . .	51
Irc::Resource . . . . .	52
ResourceData . . . . .	56
Irc::ResourceManager . . . . .	64



# Chapter 5

## Class Index

### 5.1 Class List

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

<a href="#">bz2LibCompression</a>	Compression class that uses the <i>bzip2</i> algorithm . . . . .	15
<a href="#">Collector</a>	Class to collect all resource data . . . . .	17
<a href="#">Irc::CompressDecompress</a>	Compression and Decompression base class . . . . .	21
<a href="#">CompressionFactory</a>	Factory to create compression class instances . . . . .	23
<a href="#">Irc::EncryptDecrypt</a>	Encryption and Decryption base class . . . . .	24
<a href="#">EncryptionFactory</a>	Factory to create encryption class instances . . . . .	26
<a href="#">InFileParser</a>	Base class for all <code>lrc</code> input files . . . . .	27
<a href="#">IrcEncryptionDisabledException</a>	Exception if encryption is disabled but required . . . . .	35
<a href="#">IrcFileExistsException</a>	Exception if an existing file should be overwritten . . . . .	38
<a href="#">IrcFileNotFoundException</a>	Exception class if a file could not be found . . . . .	40
<a href="#">NoneCompression</a>	Compression class that does <i>NO</i> compression . . . . .	42
<a href="#">NoneEncryption</a>	Encryption class that does <i>NO</i> encryption . . . . .	44
<a href="#">ParserFactory</a>	Factory class to create an appropriate parser class . . . . .	46
<a href="#">RCParser</a>	Class to parse an <code>.rc</code> file . . . . .	47
<a href="#">resEntry_</a>	Data for one resource entry . . . . .	51
<a href="#">Irc::Resource</a>	<a href="#">Resource</a> class for use with library . . . . .	52
<a href="#">ResourceData</a>	Internal class for resource with more information . . . . .	56
<a href="#">Irc::ResourceManager</a>	Manager class handling all resources of a resource file . . . . .	64

<a href="#">RIFParser</a>	
Class to parse a .rif (XML) file . . . . .	73
<a href="#">SerpentEncryption</a>	
Class to encrypt/decrypt using the <i>Serpent</i> algorithm . . . . .	75
<a href="#">zLibCompression</a>	
Compression class that uses the <i>zLib</i> algorithm . . . . .	78

# Chapter 6

## File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

/home/andy/Programming/Projects/lrc/src/Factoryies.hxx	90
/home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx	97
/home/andy/Programming/Projects/lrc/src/ResourceData.hxx	98
/home/andy/Programming/Projects/lrc/src/StatusCodes.hxx	99
/home/andy/Programming/Projects/lrc/src/Utils.hxx	112
/home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx	81
/home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx	82
/home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx	84
/home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx	86
/home/andy/Programming/Projects/lrc/src/compiler/RCParseer.hxx	87
/home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx	89
/home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx	91
/home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx	92
/home/andy/Programming/Projects/lrc/src/include/Resource.hxx	93
/home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx	96
/home/andy/Programming/Projects/lrc/src/strategies/bz2LibCompression.hxx	106
/home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.hxx	107
/home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx	108
/home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx	109
/home/andy/Programming/Projects/lrc/src/strategies/zLibCompression.hxx	110





## 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](#), [lastCompression](#) }
- enum [EncryptionType](#) { [NoneEncryption](#), [SerpentEncryption](#), [lastEncryption](#) }

#### 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 `liblirc` 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 CompressionType

enum `lrc::CompressionType`

Possible compression types for resource

**Todo** Add more possibilities to compress resource

Enumerator

NoneCompression	No compression at all.
zLibCompression	zlib compression
bz2LibCompression	bzip2 compression
lastCompression	Marker for last entry.

Definition at line 44 of file CompressDecompress.hxx.

### 7.1.2.2 EncryptionType

enum `lrc::EncryptionType`

Possible encryption types for resource

**Todo** Add more possibilities to encrypt resource

Enumerator

NoneEncryption	No encryption at all.
SerpentEncryption	Serpent algorithm for encryption.
lastEncryption	Marker for last entry.

Definition at line 44 of file EncryptDecrypt.hxx.

# Chapter 8

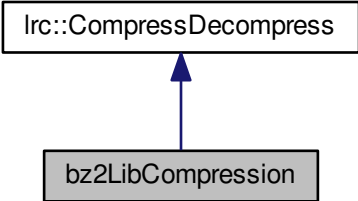
## Class Documentation

### 8.1 bz2LibCompression Class Reference

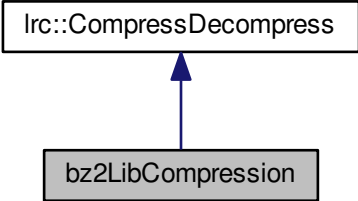
Compression class that uses the *bzip2* algorithm.

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

Inheritance diagram for bz2LibCompression:



Collaboration 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 compress()

```
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 decompress()

```
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 \*p\_rcName, char \*p\_rdfName, bool p\_overwriteAllow) throw (IrcFileExistsException)  
*Constructor.*
- [~Collector](#) (void)  
*Destructor.*
- int [collect](#) (std::vector< [ResourceData](#) \*> \*p\_resEntries, [Irc::CompressionType](#) p\_compress, [Irc::EncryptionType](#) p\_encrypt, const unsigned char \*p\_key)  
*Collect, compress and encrypt data.*

### Private Member Functions

- int [are\\_resIDs\\_unique](#) (std::vector< [ResourceData](#) \*> p\_entries, [inFilePosition](#) &p\_doubleIDPos, char \*\*p\_doubleID)  
*Check for unique resource IDs.*
- void [show\\_resource\\_data\\_error](#) (int p\_errorCode, [ResourceData](#) \*p\_resData)  
*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::Collector (
    char * p_rcName,
    char * p_rdfName,
    bool p_overwriteAllow ) 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

<a href="#"><i>IrcFileExistsException</i></a>	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::~Collector (
    void )
```

Destructor.

Clean up the memory that was needed by the class

## 8.2.3 Member Function Documentation

## 8.2.3.1 are\_resIDs\_unique()

```
int Collector::are_resIDs_unique (
    std::vector< ResourceData *> p_entries,
    inFilePosition & p_doubleIDPos,
    char ** p_doubleID ) [private]
```

Check for unique resource IDs.

This method checks if there are alle 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

## Return values

<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 collect()

```
int Collector::collect (
    std::vector< ResourceData *> * p_resEntries,
    lrc::CompressionType p_compress,
    lrc::EncryptionType p_encrypt,
    const unsigned char * p_key )
```

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 show\_resource\_data\_error()

```
void Collector::show_resource_data_error (
    int p_errorCode,
    ResourceData * p_resData ) [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>	<a href="#">ResourceData</a> instance that caused the error

## 8.2.4 Member Data Documentation

### 8.2.4.1 m\_rcName

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

Filename of .rc file.

Definition at line 50 of file Collector.hxx.

### 8.2.4.2 m\_rdfName

```
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](#)

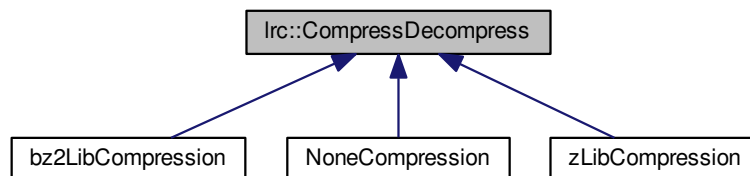


## 8.3 lrc::CompressDecompress Class Reference

Compression and Decompression base class.

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

Inheritance diagram for lrc::CompressDecompress:



### Public Member Functions

- virtual int [compress](#) (const unsigned char \*p\_decompData, size\_t p\_decompSize, unsigned char \*\*p\_compData, size\_t &p\_compSize)=0  
*Abstract method for compression.*
- virtual int [decompress](#) (const unsigned char \*p\_compData, size\_t p\_compSize, unsigned char \*\*p\_decompData, size\_t &p\_decompSize)=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 62 of file `CompressDecompress.hxx`.

### 8.3.2 Member Function Documentation

#### 8.3.2.1 compress()

```
virtual int lrc::CompressDecompress::compress (
    const unsigned char * p_decompData,
    size_t p_decompSize,
    unsigned char ** p_compData,
    size_t & p_compSize ) [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**

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

**Remarks**

The caller is responsible to free the used memory

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

**8.3.2.2 decompress()**

```
virtual int lrc::CompressDecompress::decompress (
    const unsigned char * p_compData,
    size_t p_compSize,
    unsigned char ** p_decompData,
    size_t & p_decompSize ) [pure virtual]
```

Abstract method for decompression.

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

**Parameters**

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

**Return values**

<i>NO_ERROR</i>	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/lrc/src/include/CompressDecompress.hxx

## 8.4 CompressionFactory Class Reference

Factory to create compression class instances.

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

### Static Public Member Functions

- static [lrc::CompressDecompress](#) \* [get\\_compression\\_class](#) ([lrc::CompressionType](#) p\_compType)  
*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 [lrc::CompressDecompress](#) class. The created class instance compresses and decompresses the data, depending on the given [lrc::CompressionType](#)

Definition at line 48 of file Factories.hxx.

#### 8.4.2 Member Function Documentation

##### 8.4.2.1 [get\\_compression\\_class\(\)](#)

```
static lrc::CompressDecompress* CompressionFactory::get_compression_class (  
    lrc::CompressionType p_compType ) [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:

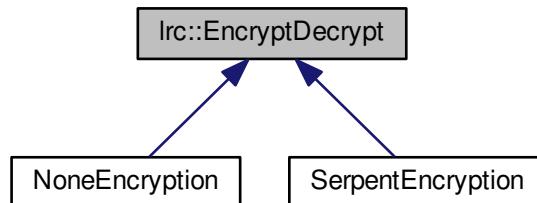
- [/home/andy/Programming/Projects/lrc/src/Factories.hxx](#)

## 8.5 Irc::EncryptDecrypt Class Reference

Encryption and Decryption base class.

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

Inheritance diagram for Irc::EncryptDecrypt:



### Public Member Functions

- virtual int `encrypt` (const unsigned char \*p\_key, const unsigned char \*p\_clearData, size\_t p\_clearSize, unsigned char \*\*p\_encData, size\_t &p\_encSize)=0  
*Abstract method for encryption.*
- virtual int `decrypt` (const unsigned char \*p\_key, const unsigned char \*p\_encData, size\_t p\_encSize, unsigned char \*\*p\_clearData, size\_t &p\_clearSize)=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 60 of file `EncryptDecrypt.hxx`.

#### 8.5.2 Member Function Documentation

##### 8.5.2.1 decrypt()

```
virtual int lrc::EncryptDecrypt::decrypt (
    const unsigned char * p_key,
    const unsigned char * p_encData,
    size_t p_encSize,
    unsigned char ** p_clearData,
    size_t & p_clearSize ) [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>	<a href="#">Resource</a> 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 encrypt()

```
virtual int lrc::EncryptDecrypt::encrypt (
    const unsigned char * p_key,
    const unsigned char * p_clearData,
    size_t p_clearSize,
    unsigned char ** p_encData,
    size_t & p_encSize ) [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>	<a href="#">Resource</a> 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/lrc/src/include/EncryptDecrypt.hxx](#)

## 8.6 EncryptionFactory Class Reference

Factory to create encryption class instances.

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

### Static Public Member Functions

- static [lrc::EncryptDecrypt](#) \* [get\\_encryption\\_class](#) ([lrc::EncryptionType](#) p\_encType, char \*p\_resID) throw ([lrc::EncryptionDisabledException](#))

*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 [lrc::EncryptDecrypt](#) class. The created class instance encrypts and decrypts the data, depending on the given [lrc::EncryptionType](#)

Definition at line 70 of file Factories.hxx.

### 8.6.2 Member Function Documentation

#### 8.6.2.1 [get\\_encryption\\_class\(\)](#)

```
static lrc::EncryptDecrypt* EncryptionFactory::get_encryption_class (  
    lrc::EncryptionType p_encType,  
    char * p_resID ) throw lrcEncryptionDisabledException    [static]
```

Creates an instance of the desired class.

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

## Parameters

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

## Returns

Instance of desired encryption class

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

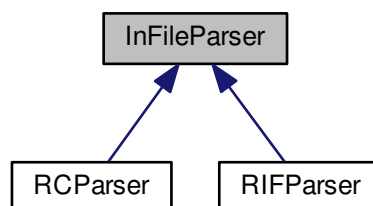
- `/home/andy/Programming/Projects/lrc/src/Factoryies.hxx`

## 8.7 InFileParser Class Reference

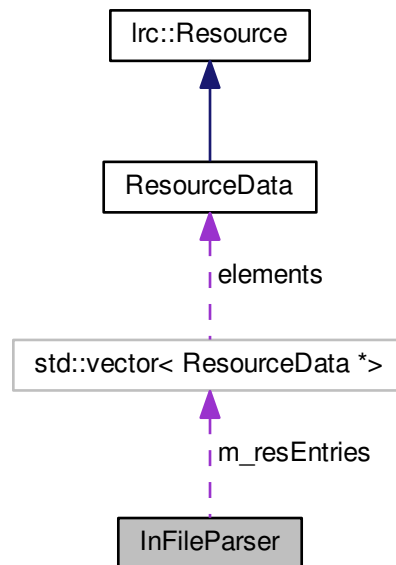
Base class for all `lrc` input files.

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

Inheritance diagram for InFileParser:



Collaboration diagram for InFileParser:



## Public Member Functions

- [InFileParser](#) (char \*p\_filename) throw (IrcFileNotFoundException)  
*Constructor.*
- [~InFileParser](#) (void)  
*Destructor.*
- virtual int [parse](#) (void)=0  
*Parses the file.*
- virtual int [get\\_internal\\_error](#) (inFilePosition &p\_errPos, char \*\*p\_errMsg)  
*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 \*p\_compStr)  
*Evaluate compression type from string.*
- static [Irc::EncryptionType eval\\_encryption\\_type](#) (const char \*p\_encStr)  
*Evaluate encryption type from string.*
- static unsigned char \* [get\\_password](#) (const char \*p\_passwdStr) throw (IrcFileNotFoundException)  
*Get password.*



## Protected Types

- enum `internalErrorType` {  
`ieNone`, `ieInvalidElement`, `ieIdentNotFound`, `ieMissingPassword`,  
`ieFilenameNotFound`, `iePasswordFileNotFound`, `ieUnknownCompression`, `ieUnknownEncryption` }

*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 internalErrorType

```
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.
ieUnknownCompression	The compression is incorrectly spelled or not available.
ieUnknownEncryption	The encryption is incorrectly spelled or not available.

Definition at line 54 of file InFileParser.hxx.

## 8.7.3 Constructor & Destructor Documentation

### 8.7.3.1 InFileParser()

```
InFileParser::InFileParser (
    char * p_filename ) throw (lrcFileNotFoundException)
```

Constructor.

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

#### Parameters

in	<i>p_filename</i>	Filename of input file
----	-------------------	------------------------

#### Exceptions

<a href="#">lrcFileNotFoundException</a>	Exception that is thrown if the given file could not be found
--	---

### 8.7.3.2 ~InFileParser()

```
InFileParser::~InFileParser (
    void )
```

Destructor.

Cleans up the memory and resources the parser class needed

## Exceptions

<a href="#">IrcFileNotFoundException</a>	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 clear\_internal\_error()

```
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 eval\_compression\_type()

```
static lrc::CompressionType InFileParser::eval_compression_type (
    const char * p_compStr ) [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

<a href="#">lrc::NoneCompression</a>	No compression
<a href="#">lrc::zLibCompression</a>	zlib compression

## Remarks

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

## 8.7.4.3 eval\_encryption\_type()

```
static lrc::EncryptionType InFileParser::eval_encryption_type (
    const char * p_encStr ) [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

<a href="#">lrc::NoneEncryption</a>	No encryption
<a href="#">lrc::SerpentEncryption</a>	Serpent encryption

## Remarks

[lrc::NoneEncryption](#) is the default if the ginve string defines no other encryption type

8.7.4.4 `get_internal_error()`

```
virtual int InFileParser::get_internal_error (
    inFilePosition & p_errPos,
    char ** p_errMsg ) [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 `get_password()`

```
static unsigned char* InFileParser::get_password (
    const char * p_passwdStr ) throw (lrcFileNotFoundException) [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

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

## Returns

Password directly from input file or from re-directed file

## Exceptions

<a href="#">IrcFileNotFoundException</a>	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 `get_resource_entries()`

```
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 `parse()`

```
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

<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

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

## 8.7.5 Member Data Documentation

### 8.7.5.1 m\_errorPosition

`inFilePosition` InFileParser::m\_errorPosition [protected]

Line and column of error.

Definition at line 67 of file InFileParser.hxx.

### 8.7.5.2 m\_filename

`char*` InFileParser::m\_filename [protected]

Filename of the RC file.

Definition at line 65 of file InFileParser.hxx.

### 8.7.5.3 m\_internalError

`internalErrorType` InFileParser::m\_internalError [protected]

Internal error.

Definition at line 69 of file InFileParser.hxx.

### 8.7.5.4 m\_lastError

`int` InFileParser::m\_lastError [protected]

Error code of last error.

Definition at line 68 of file InFileParser.hxx.

### 8.7.5.5 m\_resEntries

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

List of resource data entries.

Definition at line 66 of file InFileParser.hxx.

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

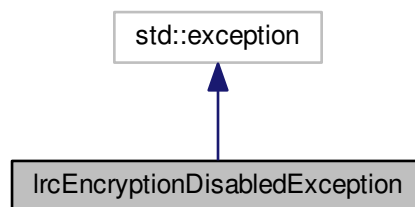
- [/home/andy/Programming/Projects/Irc/src/compiler/InFileParser.hxx](#)

## 8.8 IrcEncryptionDisabledException Class Reference

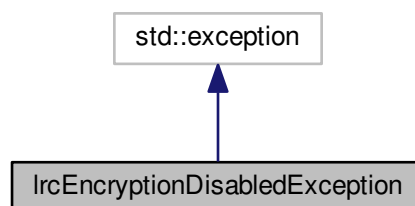
Exception if encryption is disabled but required.

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

Inheritance diagram for IrcEncryptionDisabledException:



Collaboration diagram for IrcEncryptionDisabledException:



## Public Member Functions

- [IrcEncryptionDisabledException](#) (char \*p\_resourceID)  
*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::IrcEncryptionDisabledException (
    char * p_resourceID )
```

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::~IrcEncryptionDisabledException (
    void ) throw ()
```



Destructor.

Frees the memory of the exception

### 8.8.3 Member Function Documentation

#### 8.8.3.1 what()

```
virtual const char* lrcEncryptionDisabledException::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 m\_resourceID

```
char* lrcEncryptionDisabledException::m_resourceID [private]
```

Resource that requires encryption.

Definition at line 114 of file lrcExceptions.hxx.

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

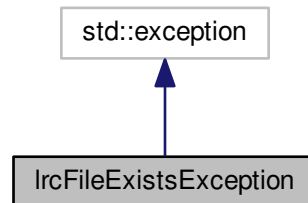
- </home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx>

## 8.9 IrcFileExistsException Class Reference

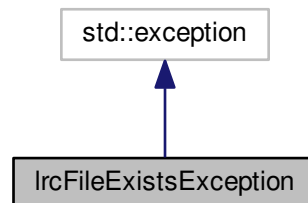
Exception if an existing file should be overwritten.

```
#include <IrcExceptions.hpp>
```

Inheritance diagram for IrcFileExistsException:



Collaboration diagram for IrcFileExistsException:



### Public Member Functions

- [IrcFileExistsException](#) (char \*p\_filename)  
*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::IrcFileExistsException (
    char * p_filename )
```

Constructor.

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

#### Parameters

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

### 8.9.2.2 ~IrcFileExistsException()

```
IrcFileExistsException::~IrcFileExistsException (
    void ) throw ()
```

Destructor.

Frees up memory needed by the class

## 8.9.3 Member Function Documentation

### 8.9.3.1 what()

```
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 m\_fileOverwrite

```
char* lrcFileExistsException::m_fileOverwrite [private]
```

Filename of file that should have been overwritten.

Definition at line 78 of file lrcExceptions.hxx.

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

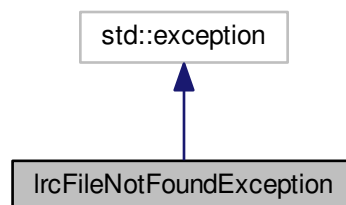
- </home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx>

## 8.10 lrcFileNotFoundExcpetion Class Reference

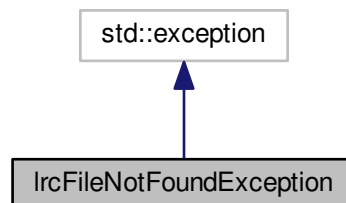
Exception class if a file could not be found.

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

Inheritance diagram for lrcFileNotFoundExcpetion:



Collaboration diagram for lrcFileNotFoundExcpetion:



## Public Member Functions

- [IrcFileNotFoundException](#) (char \*p\_fileNotFoundName)  
*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::IrcFileNotFoundException (  
    char * p_fileNotFoundName )
```

Constructor.

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

Parameters

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

#### 8.10.2.2 ~IrcFileNotFoundException()

```
IrcFileNotFoundException::~IrcFileNotFoundException (  
    void ) throw ()
```

Destructor.

### 8.10.3 Member Function Documentation

#### 8.10.3.1 what()

```
virtual const char* lrcFileNotFoundException::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 m\_fileNotFound

```
char* lrcFileNotFoundException::m_fileNotFound [private]
```

Filename of file that could not be found.

Definition at line 44 of file lrcExceptions.hxx.

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

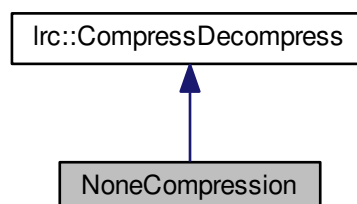
- [/home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx](#)

## 8.11 NoneCompression Class Reference

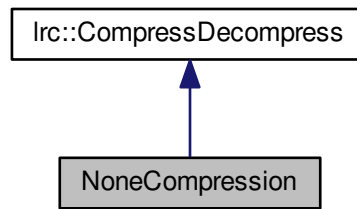
Compression class that does *NO* compression.

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

Inheritance diagram for NoneCompression:



Collaboration 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 compress()

```
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 decompress()

```
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:

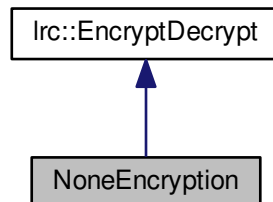
- [/home/andy/Programming/Projects/Irc/src/strategies/NoneCompression.hxx](#)

## 8.12 NoneEncryption Class Reference

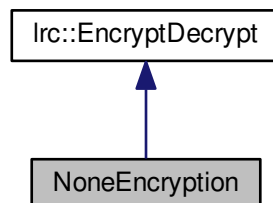
Encryption class that does *NO* encryption.

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

Inheritance diagram for NoneEncryption:



Collaboration 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 decrypt()

```
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 encrypt()

```
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/Irc/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 \*p\_filename)  
*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 create\_input\_parser()

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

in	<i>p_filename</i>	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:

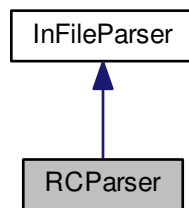
- /home/andy/Programming/Projects/lrc/src/compiler/[ParserFactory.hxx](#)

## 8.14 RCParse Class Reference

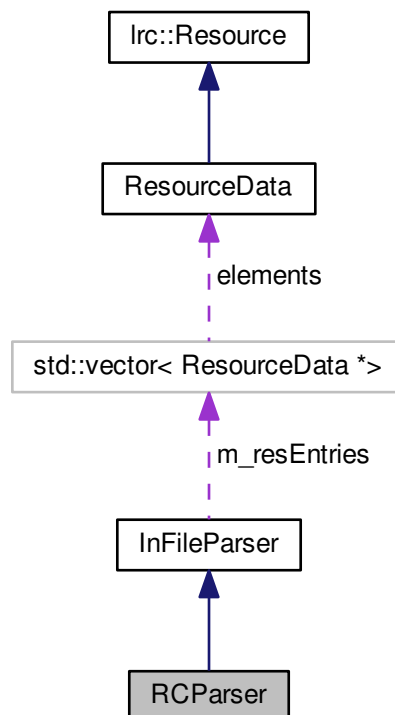
Class to parse an .rc file.

```
#include <RCParse.hpp>
```

Inheritance diagram for RCParse:



Collaboration diagram for RCParse:



## Public Member Functions

- [RCParse](#) (char \*p\_filename) throw (IrcFileNotFoundException)  
*Constructor.*
- int [parse](#) (void)  
*Parses the file.*

## Private Member Functions

- bool [is\\_comment](#) (char \*p\_line)  
*Checks if the line is a comment line.*
- bool [is\\_windows\\_line](#) (char \*p\_line)  
*Checks if the line would be valid for Windows.*
- void [copy\\_mandatory\\_data](#) (std::cmatch p\_mandatoryMatch, bool p\_winData, char \*\*p\_resident, char \*\*p\_resFilename)  
*Copy mandatory data in existing variables.*

## 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 44 of file RCParse.hxx.

### 8.14.2 Constructor & Destructor Documentation

#### 8.14.2.1 RCParse()

```
RCParse::RCParse (
    char * p_filename ) 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 RC file
----	-------------------	---------------------

## Exceptions

<a href="#">IrcFileNotFoundException</a>	Exception that is thrown if the given .rc file could not be found
--	---

## 8.14.3 Member Function Documentation

## 8.14.3.1 copy\_mandatory\_data()

```
void RCParse::copy_mandatory_data (
    std::cmatch p_mandatoryMatch,
    bool p_winData,
    char ** p_resIdent,
    char ** p_resFilename ) [private]
```

Copy mandatory data in existing variables.

This method copies the mandatory data from a C++ regex match into the existing variables. It differs between a Windows and a normal Irc match

## Parameters

in	<i>p_mandatoryMatch</i>	C++ match containing the data
in	<i>p_winData</i>	Flag indicating windows
out	<i>p_resIdent</i>	Resource identifier
out	<i>p_resFilename</i>	Resource filename

## 8.14.3.2 is\_comment()

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

Checks if the line is a comment line.

This method reads the given text as a text line from left to right and 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

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

## Return values

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

### 8.14.3.3 is\_windows\_line()

```
bool RCParse::is_windows_line (
    char * p_line ) [private]
```

Checks if the line would be valid for Windows.

This method checks if the given line is a valid line for a Windows RC file. It will be called whenever the check for the mandatory lrc entries fail.

#### Parameters

in	<i>p_line</i>	Line of RC file
----	---------------	-----------------

#### Return values

<i>true</i>	It is a valid line for Windows RC
<i>false</i>	It would also fail on Windows

### 8.14.3.4 parse()

```
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.*
- [lrc::EncryptionType](#) [encType](#)  
*Type of encryption of this entry.*
- [lrc::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 93 of file Resource.hxx.

### 8.15.2 Member Data Documentation

#### 8.15.2.1 compType

```
lrc::CompressionType resEntry_::compType
```

Type of compression of this entry.

Definition at line 98 of file Resource.hxx.

#### 8.15.2.2 encType

```
lrc::EncryptionType resEntry_::encType
```

Type of encryption of this entry.

Definition at line 97 of file Resource.hxx.

### 8.15.2.3 resID

```
char resEntry_::resID[MAX_ID_LEN]
```

Resource ID.

Definition at line 94 of file Resource.hxx.

### 8.15.2.4 resSize

```
unsigned int resEntry_::resSize
```

Size of resource.

Definition at line 96 of file Resource.hxx.

### 8.15.2.5 startOffset

```
unsigned int resEntry_::startOffset
```

Offset from the start.

Definition at line 95 of file Resource.hxx.

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

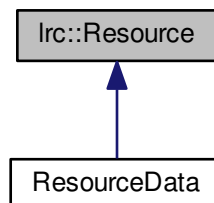
- [/home/andy/Programming/Projects/lrc/src/include/Resource.hxx](#)

## 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 127 of file Resource.hxx.

### 8.16.2 Constructor & Destructor Documentation

#### 8.16.2.1 Resource()

```
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 ~Resource()

```
lrc::Resource::~~Resource (
    void )
```

Destructor.

Cleans up the used memory of the class

## 8.16.3 Member Function Documentation

### 8.16.3.1 get\_ID()

```
char* lrc::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 get\_res\_data()

```
unsigned char* lrc::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 get\_res\_size()

```
size_t lrc::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 m\_resData

```
unsigned char* lrc::Resource::m_resData [protected]
```

Pointer to resource data.

Definition at line 131 of file Resource.hxx.

### 8.16.4.2 m\_resID

```
char* lrc::Resource::m_resID [protected]
```

Resource ID.

Definition at line 130 of file Resource.hxx.

### 8.16.4.3 m\_resSize

```
size_t lrc::Resource::m_resSize [protected]
```

Size of resource data.

Definition at line 132 of file Resource.hxx.

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

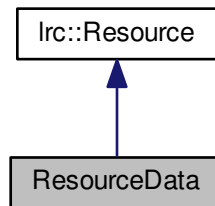
- /home/andy/Programming/Projects/lrc/src/include/[Resource.hxx](#)

## 8.17 ResourceData Class Reference

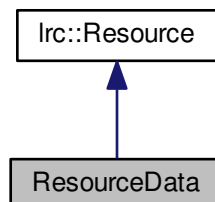
Internal class for resource with more information.

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

Inheritance diagram for ResourceData:



Collaboration diagram for ResourceData:



### Public Member Functions

- [ResourceData](#) (void)  
*Constructor.*
- [~ResourceData](#) (void)  
*Destructor.*
- void [set\\_ident](#) (const char \*p\_resident)  
*Set resource identifier.*
- void [set\\_file](#) (const char \*p\_resFilename)  
*Set file containing resource.*
- char \* [get\\_file](#) (void)  
*Return filename.*
- void [set\\_encryption](#) ([Irc::EncryptionType](#) p\_resEncryption, const unsigned char \*p\_password)

- Set encryption type.*

  - [Irc::EncryptionType get\\_encryption](#) (void)

*Return encryption type.*
- void [set\\_compression](#) ([Irc::CompressionType](#) p\_resCompression)

*Set compression type.*
- [Irc::CompressionType get\\_compression](#) (void)

*Return compression type.*
- void [set\\_rc\\_position](#) (int p\_line, int p\_col)

*Set position of resource description.*
- [inFilePosition get\\_rc\\_position](#) (void)

*Return resource description position.*
- int [prepare\\_resource\\_from\\_file](#) (unsigned char \*\*p\_resData, size\_t &p\_resSize)

*Prepare resource for collecting.*
- int [get\\_data\\_from\\_memory](#) (unsigned char \*p\_dataStart, [resEntry](#) p\_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 \*p\_newErrMsg)

*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::ResourceData (
    void )
```

Constructor.

Standard constructor to initialize the class

### 8.17.2.2 ~ResourceData()

```
ResourceData::~~ResourceData (
    void )
```

Destructor.

Cleans up memory used from the class

## 8.17.3 Member Function Documentation

### 8.17.3.1 get\_compression()

```
lrc::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 get\_data\_from\_memory()

```
int ResourceData::get_data_from_memory (
    unsigned char * p_dataStart,
    resEntry p_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 `get_encryption()`

```
lrc::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 `get_error_msg()`

```
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 get\_file()

```
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 get\_rc\_position()

```
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 prepare\_resource\_from\_file()

```
int ResourceData::prepare_resource_from_file (
    unsigned char ** p_resData,
    size_t & p_resSize )
```

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



## Return values

<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 set\_compression()

```
void ResourceData::set_compression (
    lrc::CompressionType p_resCompression )
```

Set compression type.

Method to set compression type of resource

## Parameters

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

## 8.17.3.9 set\_encryption()

```
void ResourceData::set_encryption (
    lrc::EncryptionType p_resEncryption,
    const unsigned char * p_password )
```

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 set\_error\_msg()

```
void ResourceData::set_error_msg (
    char * p_newErrMsg ) [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 set\_file()

```
void ResourceData::set_file (
    const char * p_resFilename )
```

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 set\_ident()

```
void ResourceData::set_ident (
    const char * p_resIdent )
```

Set resource identifier.

Method to set the internal resource identifier

#### Parameters

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

### 8.17.3.13 set\_rc\_position()

```
void ResourceData::set_rc_position (
```

```
int p_line,  
int p_col )
```

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 m\_compression

```
lrc::CompressionType ResourceData::m_compression [protected]
```

Type of compression for this resource.

Definition at line 61 of file ResourceData.hxx.

### 8.17.4.2 m\_encryption

```
lrc::EncryptionType ResourceData::m_encryption [protected]
```

Type of encryption for this resource.

Definition at line 60 of file ResourceData.hxx.

### 8.17.4.3 m\_errorMsg

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

Error message (if any)

Definition at line 64 of file ResourceData.hxx.

#### 8.17.4.4 m\_filename

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

Filename of file containing resource.

Definition at line 59 of file ResourceData.hxx.

#### 8.17.4.5 m\_inFilePosition

```
inFilePosition ResourceData::m_inFilePosition [protected]
```

Position of resource in RC file.

Definition at line 62 of file ResourceData.hxx.

#### 8.17.4.6 m\_password

```
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:

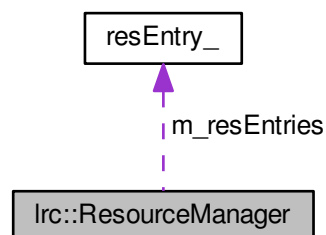
- [/home/andy/Programming/Projects/lrc/src/ResourceData.hxx](#)

## 8.18 Irc::ResourceManager Class Reference

Manager class handling all resources of a resource file.

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

Collaboration diagram for Irc::ResourceManager:



## Public Member Functions

- [ResourceManager](#) (const char \*p\_resFilename, const [CompressionType](#) p\_compress, const [EncryptionType](#) p\_encrypt, const unsigned char \*p\_key) throw (IrcFileNotFoundException)  
*Constructor expecting resource file name.*
- [ResourceManager](#) (const unsigned char \*p\_startAddr, const unsigned char \*p\_endAddr, const [CompressionType](#) p\_compress, const [EncryptionType](#) p\_encrypt, const unsigned char \*p\_key)  
*Constructor for embedded resources.*
- [~ResourceManager](#) (void)  
*Destructor.*
- char \*\* [get\\_resource\\_ids](#) (int &p\_numRes)  
*Returns a list of all resources by ID.*
- [Resource \\*](#) [get\\_resource](#) (const char \*p\_resID, const unsigned char \*p\_password=nullptr)  
*Returns the requested resource.*
- [Resource \\*](#) [get\\_resource](#) (unsigned int p\_resIdx, const unsigned char \*p\_password=nullptr)  
*Returns the resource at the index.*

## Private Member Functions

- int [decrypt\\_data](#) (const unsigned char \*p\_encData, size\_t p\_encSize, unsigned char \*\*p\_clearData, size\_t &p\_clearSize)  
*Decrypt resource data.*
- int [decompress\\_data](#) (const unsigned char \*p\_compData, size\_t p\_compSize, unsigned char \*\*p\_decompData, size\_t &p\_decompSize)  
*Decompress resource data.*
- int [setup\\_resources](#) (const unsigned char \*p\_resData, size\_t p\_resSize)  
*Setup new resources.*
- int [load\\_from\\_file](#) (void)  
*Method to load resource file.*
- int [load\\_embedded](#) (const unsigned char \*p\_startAddr, const unsigned char \*p\_endAddr)  
*Method to load embedded resource.*

## 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. If the constructor with no filename is used, it expects the resource data is directly compiled into the executable file.

Definition at line 52 of file `ResourceManager.hxx`.

### 8.18.2 Constructor & Destructor Documentation

#### 8.18.2.1 `ResourceManager()` [1/2]

```
lrc::ResourceManager::ResourceManager (
    const char * p_resFilename,
    const CompressionType p_compress,
    const EncryptionType p_encrypt,
    const unsigned char * p_key ) throw (lrcFileNotFoundException)
```

Constructor expecting resource file name.

The constructor expects the name of a resource file. The file will be loaded at first use, but an exception 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

<a href="#">lrcFileNotFoundException</a>	Will be thrown if the file does not exist
--	---

#### 8.18.2.2 `ResourceManager()` [2/2]

```
lrc::ResourceManager::ResourceManager (
    const unsigned char * p_startAddr,
    const unsigned char * p_endAddr,
    const CompressionType p_compress,
    const EncryptionType p_encrypt,
    const unsigned char * p_key )
```

Constructor for embedded resources.

The constructor expects the start and end address of the resource data in memory and the compression and encryption type if the complete resource is compressed and encrypted respectively. If it is encrypted a password is expected.

#### Parameters

in	<i>p_startAddr</i>	Start address of resource data in memory
in	<i>p_endAddr</i>	End address of resource data in memory
in	<i>p_compress</i>	Compression type if the complete resource data is compressed
in	<i>p_encrypt</i>	Encryption type if the complete resource data is encrypted
in	<i>p_key</i>	Password if the complete resource is encrypted

#### Remarks

How to get to the start and end address in the linked file, see the [Wiki](#) or the man page

#### 8.18.2.3 ~ResourceManager()

```
lrc::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 decompress\_data()

```
int lrc::ResourceManager::decompress_data (
    const unsigned char * p_compData,
    size_t p_compSize,
    unsigned char ** p_decompData,
    size_t & p_decompSize ) [private]
```

Decompress resource data.

This method decompresses the resource data in the memory

#### Parameters

in	<i>p_compData</i>	Compressed resource data
in	<i>p_compSize</i>	Size of compressed resource data
out	<i>p_decompData</i>	Decompressed resource data
Generated by Doxygen	<i>p_decompSize</i>	Size of decompressed resource data

## Return values

<i>NO_ERROR</i>	Data successfully decompressed
<i>ERROR_COMPRESSION_DECOMPRESS</i>	An error occurred while decompressing the data

8.18.3.2 `decrypt_data()`

```
int lrc::ResourceManager::decrypt_data (
    const unsigned char * p_encData,
    size_t p_encSize,
    unsigned char ** p_clearData,
    size_t & p_clearSize ) [private]
```

Decrypt resource data.

This methods decrypts the resource data in the memory

## Parameters

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>	Data successfully decrypted
<i>ERROR_ENCRYPTION_NOT_AVAILABLE</i>	Encryption/Decryption support is not compiled in
<i>ERROR_ENCRYPTION_DECRYPT</i>	An error occurred while decrypting the data

8.18.3.3 `get_resource()` [1/2]

```
Resource* lrc::ResourceManager::get_resource (
    const char * p_resID,
    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

**8.18.3.4 get\_resource()** [2/2]

```
Resource* lrc::ResourceManager::get_resource (
    unsigned int p_resIdx,
    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

**8.18.3.5 get\_resource\_ids()**

```
char** lrc::ResourceManager::get_resource_ids (
    int & p_numRes )
```

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

## 8.18.3.6 load\_embedded()

```
int lrc::ResourceManager::load_embedded (
    const unsigned char * p_startAddr,
    const unsigned char * p_endAddr ) [private]
```

Method to load embedded resource.

This method loads all embedded resource data

## Parameters

in	<i>p_startAddr</i>	Start address of resource data in memory
in	<i>p_endAddr</i>	End address of resource data in memory

## Return values

<i>NO_ERROR</i>	Embedded resources successfully accessed
<i>ERROR_ENCRYPTION_NOT_AVAILABLE</i>	Encryption/Decryption support is not compiled in
<i>ERROR_ENCRYPTION_DECRYPT</i>	An error occurred while decrypting the data
<i>ERROR_COMPRESSION_DECOMPRESS</i>	An error occurred while decompressing the data

## 8.18.3.7 load\_from\_file()

```
int lrc::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>	<a href="#">Resource</a> file successfully loaded
<i>ERROR_FILE_READ</i>	An error occurred while reading the file
<i>ERROR_ENCRYPTION_NOT_AVAILABLE</i>	Encryption/Decryption support is not compiled in
<i>ERROR_ENCRYPTION_DECRYPT</i>	An error occurred while decrypting the data
<i>ERROR_COMPRESSION_DECOMPRESS</i>	An error occurred while decompressing the data

### 8.18.3.8 setup\_resources()

```
int lrc::ResourceManager::setup_resources (
    const unsigned char * p_resData,
    size_t p_resSize ) [private]
```

Setup new resources.

This method sets up internal data to provide the resource data in the memory

#### Parameters

in	<i>p_resData</i>	Decrypted and decompressed resource data
in	<i>p_resSize</i>	Size of resource data

#### Return values

<i>NO_ERROR</i>	Internal data successfully set up
-----------------	-----------------------------------

## 8.18.4 Member Data Documentation

### 8.18.4.1 m\_compType

```
CompressionType lrc::ResourceManager::m_compType [private]
```

Compression type for complete file.

Definition at line 60 of file ResourceManager.hxx.

### 8.18.4.2 m\_encType

```
EncryptionType lrc::ResourceManager::m_encType [private]
```

Encryption type for complete file.

Definition at line 61 of file ResourceManager.hxx.

### 8.18.4.3 m\_numResEntries

```
int lrc::ResourceManager::m_numResEntries [private]
```

Number of entries in resource file.

Definition at line 57 of file ResourceManager.hxx.

#### 8.18.4.4 m\_password

```
unsigned char* lrc::ResourceManager::m_password [private]
```

Password is complete file is encrypted.

Definition at line 62 of file ResourceManager.hxx.

#### 8.18.4.5 m\_resData

```
unsigned char* lrc::ResourceManager::m_resData [private]
```

Pointer to data of resource file.

Definition at line 58 of file ResourceManager.hxx.

#### 8.18.4.6 m\_resDataSize

```
size_t lrc::ResourceManager::m_resDataSize [private]
```

Size of resource data.

Definition at line 59 of file ResourceManager.hxx.

#### 8.18.4.7 m\_resEntries

```
resEntry* lrc::ResourceManager::m_resEntries [private]
```

Entries of the resource file.

Definition at line 56 of file ResourceManager.hxx.

#### 8.18.4.8 m\_resourceFile

```
char* lrc::ResourceManager::m_resourceFile [private]
```

Filename of resource file.

Definition at line 55 of file ResourceManager.hxx.

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

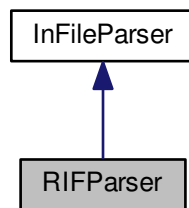
- [/home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx](#)

## 8.19 RIFParser Class Reference

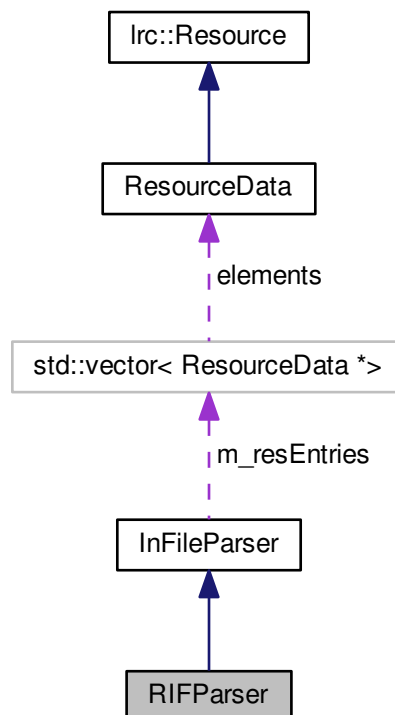
Class to parse a .rif (XML) file.

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

Inheritance diagram for RIFParser:



Collaboration diagram for RIFParser:



## Public Member Functions

- [RIFParser](#) (char \*p\_filename) 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::RIFParser (
    char * p_filename ) 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

<a href="#">IrcFileNotFoundException</a>	Exception that is thrown if the given .rif file could not be found
--	--

### 8.19.3 Member Function Documentation

#### 8.19.3.1 parse()

```
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

<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:

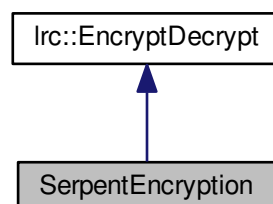
- `/home/andy/Programming/Projects/Irc/src/compiler/RIFParser.hxx`

## 8.20 SerpentEncryption Class Reference

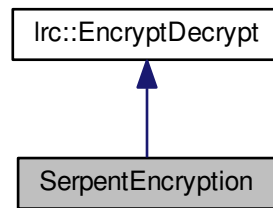
Class to encrypt/decrypt using the *Serpent* algorithm.

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

Inheritance diagram for SerpentEncryption:



Collaboration diagram for SerpentEncryption:



### Public Member Functions

- int [encrypt](#) (const unsigned char \*p\_key, const unsigned char \*p\_clearData, size\_t p\_clearSize, unsigned char \*\*p\_encData, size\_t &p\_encSize)  
*Serpent encryption.*
- int [decrypt](#) (const unsigned char \*p\_key, const unsigned char \*p\_encData, size\_t p\_encSize, unsigned char \*\*p\_clearData, size\_t &p\_clearSize)  
*Serpent decryption.*

### Protected Member Functions

- int [create\\_initialization\\_vector](#) (unsigned char \*p\_ivArray, size\_t p\_ivSize)  
*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 create\_initialization\_vector()

```
int SerpentEncryption::create_initialization_vector (
    unsigned char * p_ivArray,
    size_t p_ivSize ) [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 decrypt()

```
int SerpentEncryption::decrypt (
    const unsigned char * p_key,
    const unsigned char * p_encData,
    size_t p_encSize,
    unsigned char ** p_clearData,
    size_t & p_clearSize ) [virtual]
```

Serpent decryption.

Method to decrypt the given data using the Serpent algorithm

Implements [Irc::EncryptDecrypt](#).

## 8.20.2.3 encrypt()

```
int SerpentEncryption::encrypt (
    const unsigned char * p_key,
    const unsigned char * p_clearData,
    size_t p_clearSize,
    unsigned char ** p_encData,
    size_t & p_encSize ) [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:

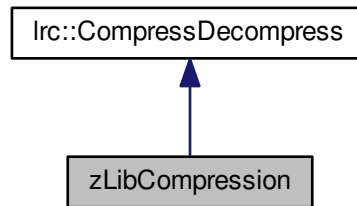
- [/home/andy/Programming/Projects/Irc/src/strategies/SerpentEncryption.hxx](#)

## 8.21 zLibCompression Class Reference

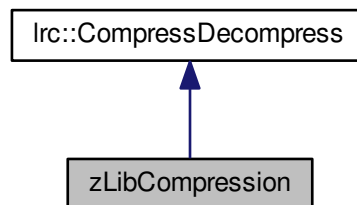
Compression class that uses the *zLib* algorithm.

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

Inheritance diagram for zLibCompression:



Collaboration 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 compress()

```
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 decompress()

```
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/lrc/src/strategies/zLibCompression.hxx](#)

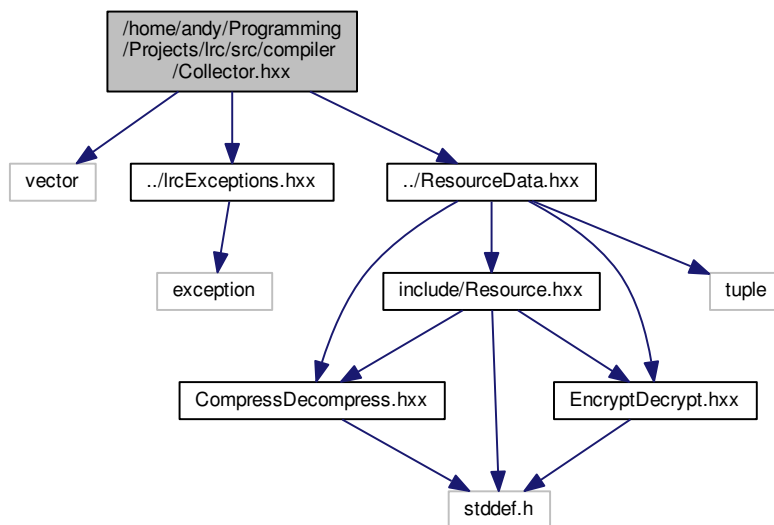


## Chapter 9

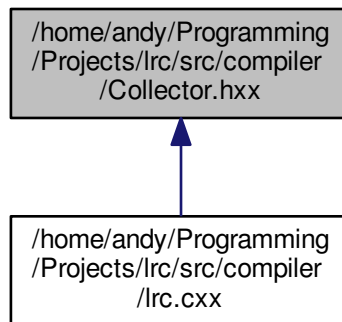
# File Documentation

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

```
#include <vector>
#include "../lrcExceptions.hxx"
#include "../ResourceData.hxx"
Include dependency graph for Collector.hxx:
```



This graph shows which files directly or indirectly include this file:



## 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 Tscharner

#### Date

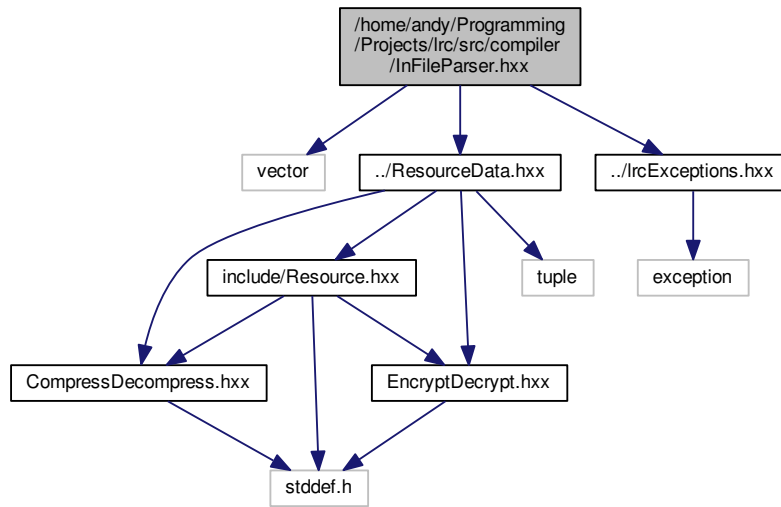
2014-08-23

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

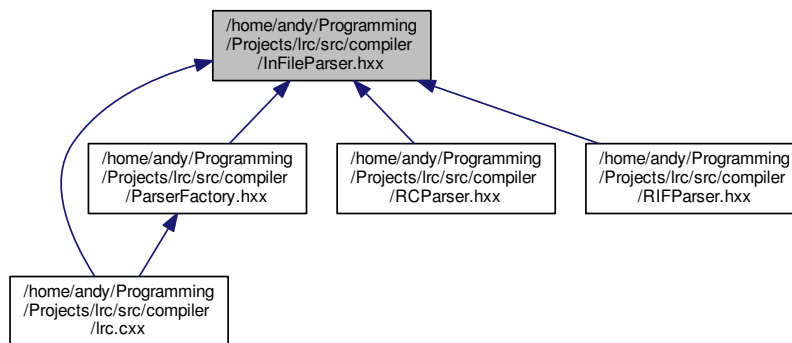
```
#include <vector>
#include "../ResourceData.hxx"
```

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

Include dependency graph for InFileParser.hxx:



This graph shows which files directly or indirectly include this file:



## 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**

2014-10-26

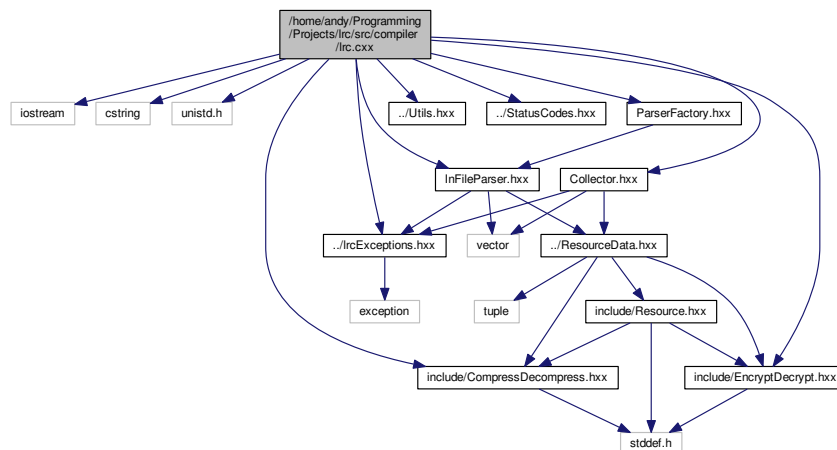
**9.3 /home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx File Reference**

```

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

```

Include dependency graph for lrc.cxx:

**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 `convert_to_elf` (const char \*p\_rdfFilename, const char \*p\_elfFilename)  
Convert data to ELF binary format.
- 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 Tschärner

#### Date

2014-08-23

### 9.3.2 Macro Definition Documentation

#### 9.3.2.1 VERSION

```
#define VERSION "<undefined>"
```

Define for version number if generated header file is missing.

Definition at line 46 of file lrc.cxx.

### 9.3.3 Function Documentation

#### 9.3.3.1 convert\_to\_elf()

```
int convert_to_elf (
    const char * p_rdfFilename,
    const char * p_elfFilename )
```

Convert data to ELF binary format.

Definition at line 72 of file lrc.cxx.

#### 9.3.3.2 main()

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

Main program.

Definition at line 103 of file lrc.cxx.

### 9.3.3.3 usage()

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

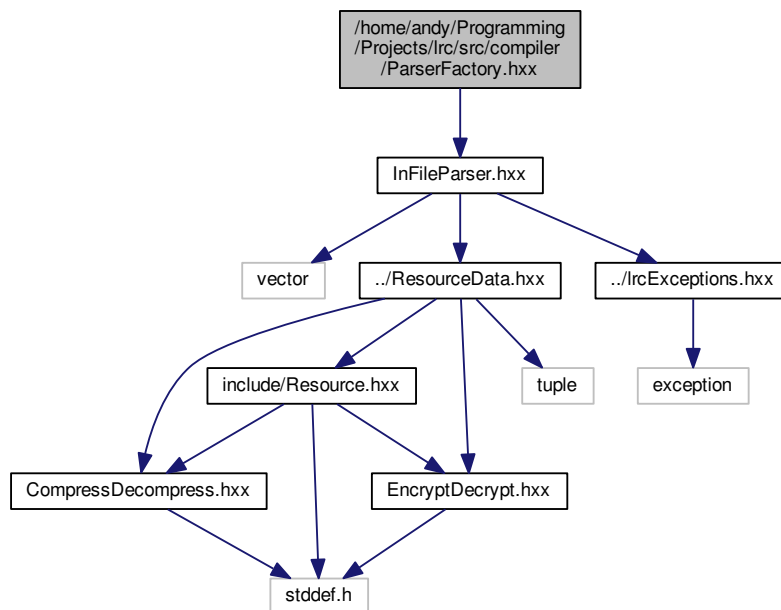
Show the usage of the compiler.

Definition at line 55 of file lrc.cxx.

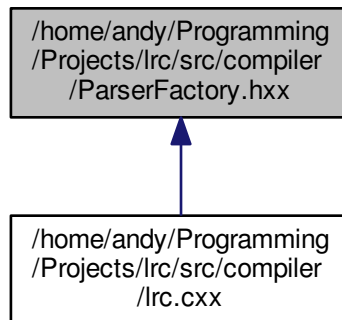
## 9.4 /home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx File Reference

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

Include dependency graph for ParserFactory.hxx:



This graph shows which files directly or indirectly include this file:



## 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 Tschärner

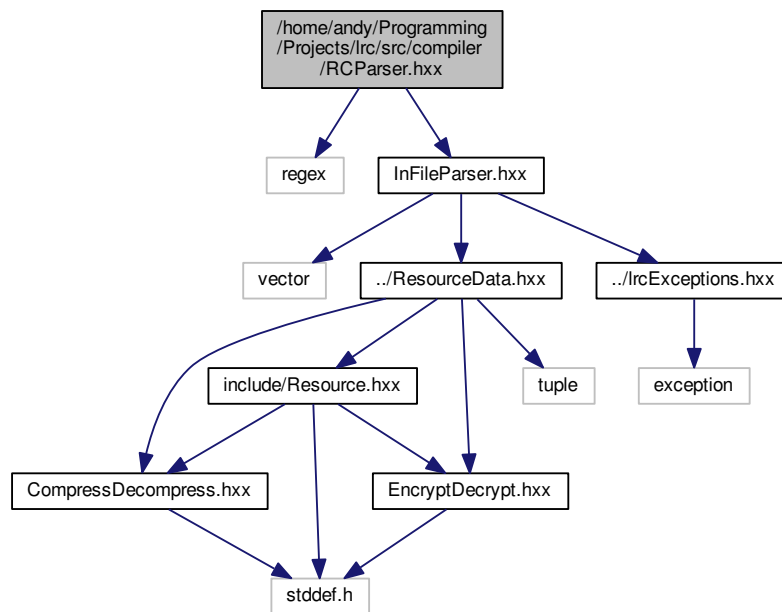
#### Date

2014-08-23

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

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

Include dependency graph for RCParse.hxx:



## Classes

- class [RCParse](#)

*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 Tschärner

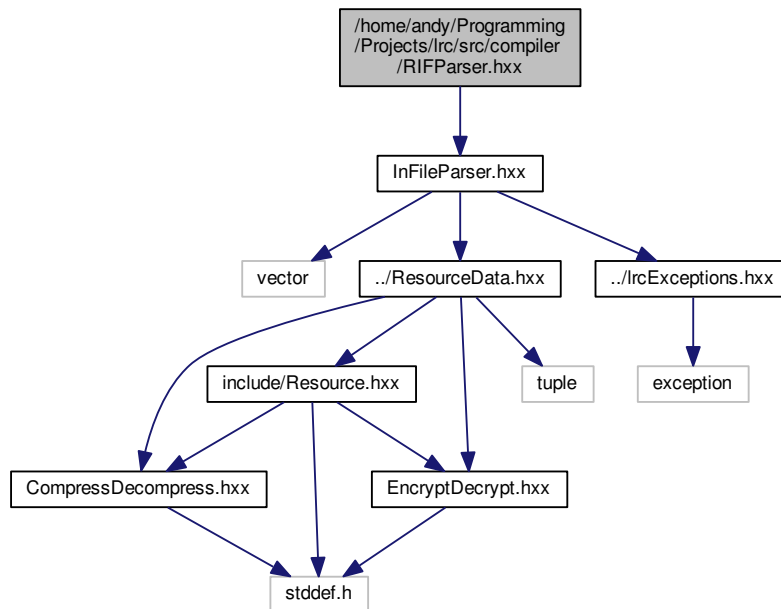
#### Date

2014-10-26

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

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

Include dependency graph for RIFParser.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 Tschärner

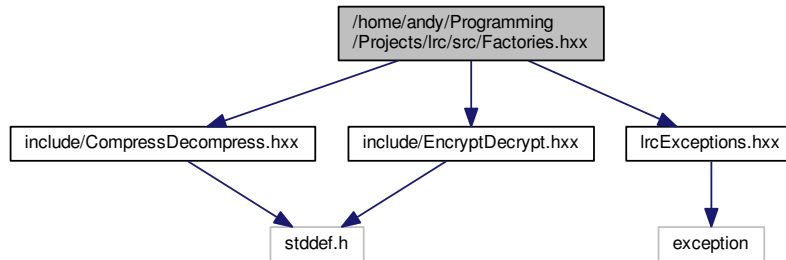
#### Date

2014-08-23

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

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

Include dependency graph for Factories.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 Tschärner

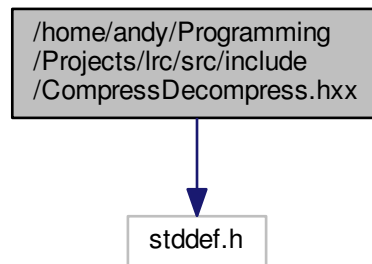
#### Date

2014-08-23

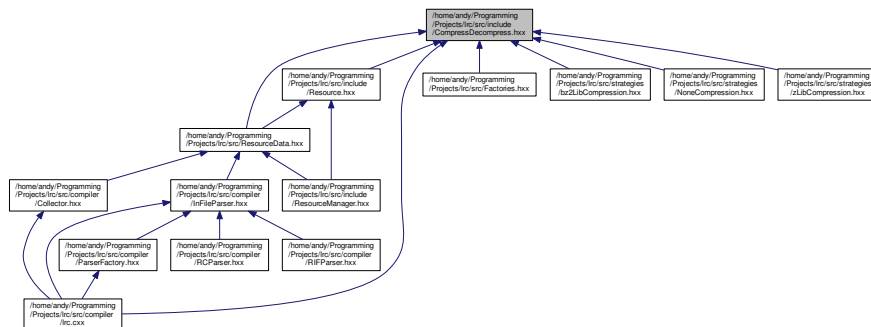
## 9.8 /home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx File Reference

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

Include dependency graph for CompressDecompress.hxx:



This graph shows which files directly or indirectly include this file:



### Classes

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

### Namespaces

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

### Enumerations

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

### 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

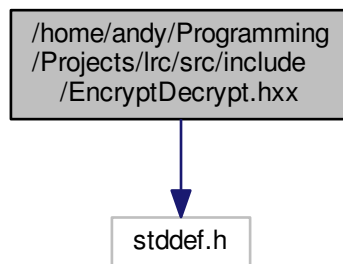
#### Date

2014-08-23

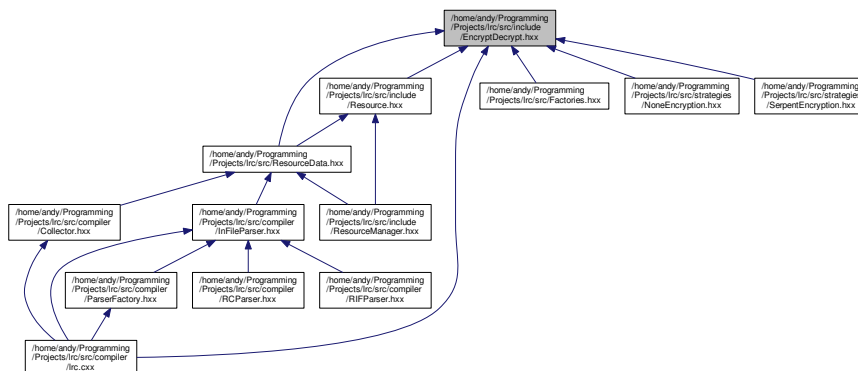
## 9.9 `/home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx` File Reference

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

Include dependency graph for `EncryptDecrypt.hxx`:



This graph shows which files directly or indirectly include this file:





## Classes

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

## Namespaces

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

## Enumerations

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

### 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

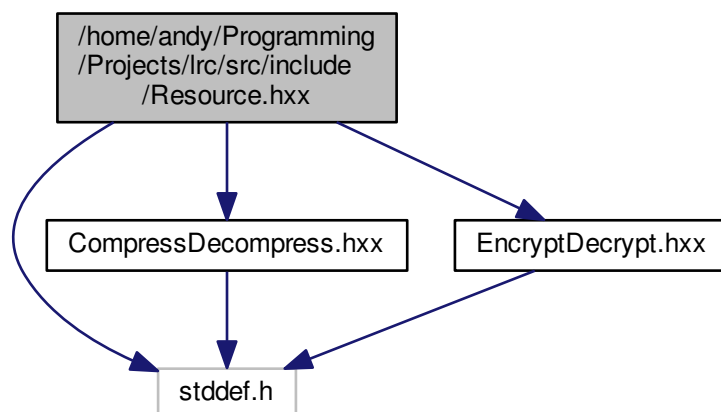
Andreas Tschärner

#### Date

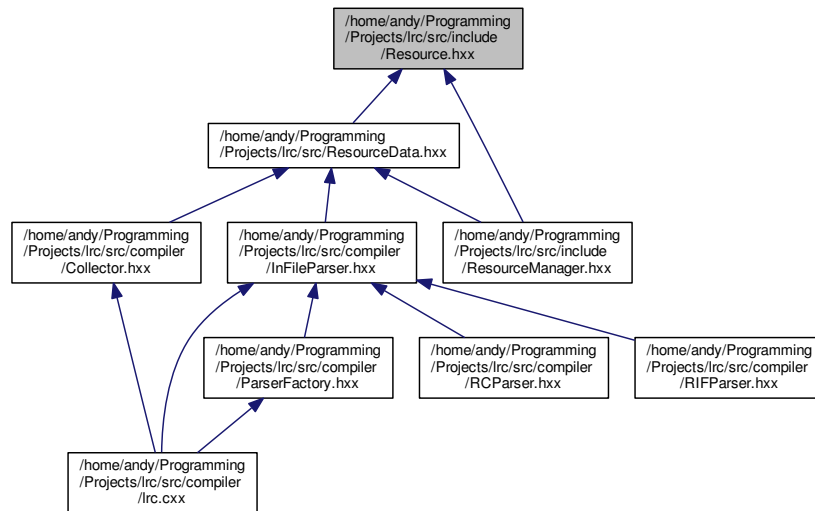
2014-08-23

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

```
#include <stddef.h>
#include "CompressDecompress.hxx"
#include "EncryptDecrypt.hxx"
Include dependency graph for Resource.hxx:
```



This graph shows which files directly or indirectly include this file:



## Classes

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

## Namespaces

- [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 Tschärner

#### Date

2014-08-22

### 9.10.2 Macro Definition Documentation

#### 9.10.2.1 MAX\_ID\_LEN

```
#define MAX_ID_LEN 80
```

Maximum length of resource ID.

Definition at line 85 of file Resource.hxx.

### 9.10.3 Typedef Documentation

#### 9.10.3.1 resEntry

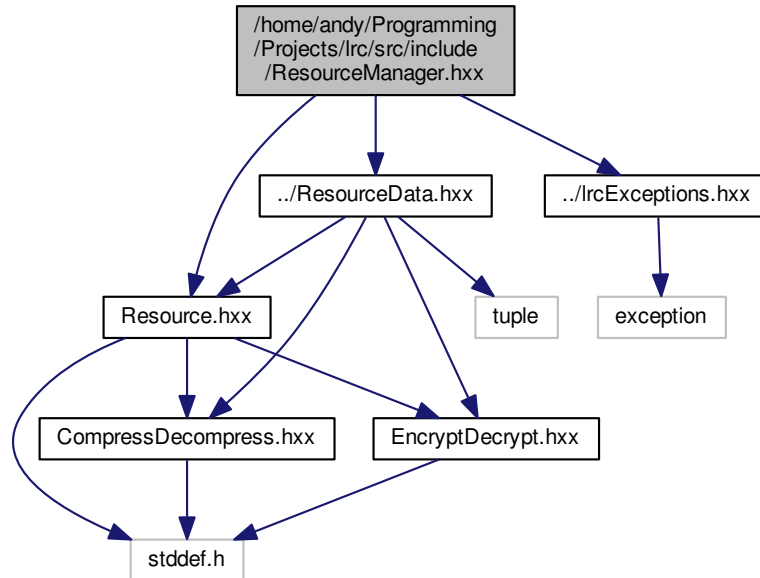
```
typedef struct resEntry_ resEntry
```

Define a better name for struct `resEntry_`.

Definition at line 102 of file Resource.hxx.

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

```
#include "Resource.hxx"
#include "../ResourceData.hxx"
#include "../lrcExceptions.hxx"
Include dependency graph for ResourceManager.hxx:
```



### Classes

- class `lrc::ResourceManager`  
*Manager class handling all resources of a resource file.*

### Namespaces

- `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

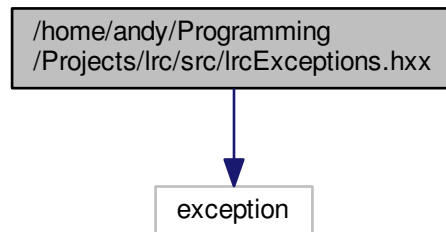
#### Date

2014-08-23

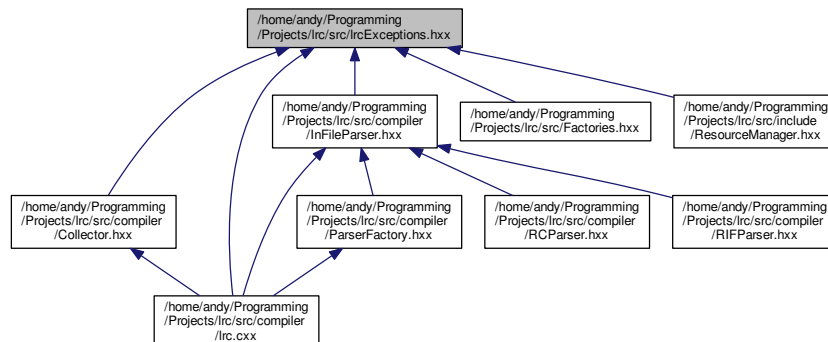
## 9.12 /home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx File Reference

```
#include <exception>
```

Include dependency graph for lrcExceptions.hxx:



This graph shows which files directly or indirectly include this file:



### 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

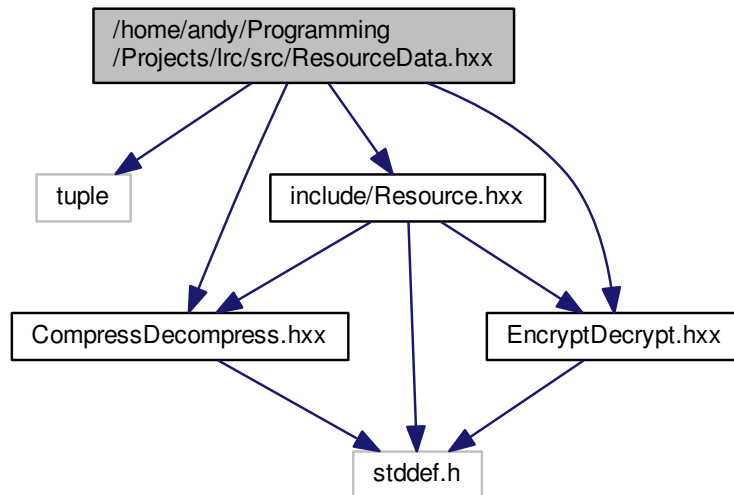
Andreas Tschärner

## Date

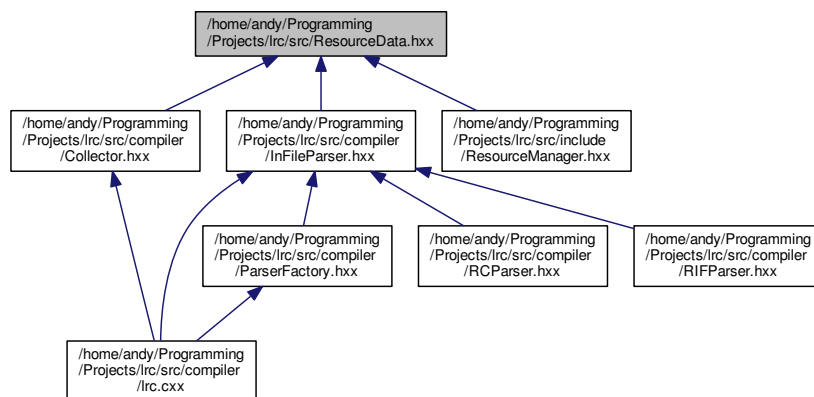
2014-08-23

## 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"
Include dependency graph for ResourceData.hxx:
```



This graph shows which files directly or indirectly include this file:



## 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 Tschärner

#### Date

2014-08-023

### 9.13.2 Typedef Documentation

#### 9.13.2.1 inFilePosition

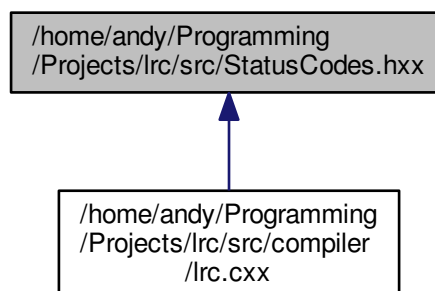
```
typedef std::tuple<int, int> inFilePosition
```

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

This graph shows which files directly or indirectly include this file:



## 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_UNKNOWN_ARCH ERROR_BASE-40`  
*The architecture is unknown.*
- `#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

2014-08-10

### 9.14.2 Macro Definition Documentation

#### 9.14.2.1 ERROR\_BASE

```
#define ERROR_BASE -1000
```

Base value for all error codes.

Definition at line 44 of file StatusCodes.hxx.

#### 9.14.2.2 ERROR\_COMPRESSION\_COMPRESS

```
#define ERROR_COMPRESSION_COMPRESS ERROR_BASE-22
```

An error occurred while compressing.

Definition at line 58 of file StatusCodes.hxx.

#### 9.14.2.3 ERROR\_COMPRESSION\_DECOMPRESS

```
#define ERROR_COMPRESSION_DECOMPRESS ERROR_BASE-23
```

An error occurred while decompressing.

Definition at line 59 of file StatusCodes.hxx.

#### 9.14.2.4 ERROR\_COMPRESSION\_NOT\_AVAILABLE

```
#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 ERROR\_ENCRYPTION\_DECRYPT

```
#define ERROR_ENCRYPTION_DECRYPT ERROR_BASE-25
```

An error occurred while decrypting.

Definition at line 61 of file StatusCodes.hxx.

#### 9.14.2.6 ERROR\_ENCRYPTION\_ENCRYPT

```
#define ERROR_ENCRYPTION_ENCRYPT ERROR_BASE-24
```

An error occurred while encrypting.

Definition at line 60 of file StatusCodes.hxx.

#### 9.14.2.7 ERROR\_ENCRYPTION\_NOT\_AVAILABLE

```
#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 ERROR\_FILE\_NOT\_FOUND

```
#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 ERROR\_FILE\_OPEN

```
#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 ERROR\_FILE\_READ

```
#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 ERROR\_FILE\_WRITE

```
#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 ERROR\_INVALID\_PARAMETER

```
#define ERROR_INVALID_PARAMETER ERROR_BASE-100
```

The provided parameter was illegal.

Definition at line 67 of file StatusCodes.hxx.

#### 9.14.2.13 ERROR\_NOT\_ENOUGH\_MEMORY

```
#define ERROR_NOT_ENOUGH_MEMORY ERROR_BASE-101
```

Not enough memory available for this action.

Definition at line 68 of file StatusCodes.hxx.

#### 9.14.2.14 ERROR\_PARSE

```
#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 ERROR\_UNKNOWN\_ARCH

```
#define ERROR_UNKNOWN_ARCH ERROR_BASE-40
```

The architecture is unknown.

Definition at line 64 of file StatusCodes.hxx.

#### 9.14.2.16 NO\_ERROR

```
#define NO_ERROR 0
```

No error, everything OK.

Definition at line 37 of file StatusCodes.hxx.

#### 9.14.2.17 WARNING\_BASE

```
#define WARNING_BASE 0
```

Base value for all warning codes.

Definition at line 40 of file StatusCodes.hxx.

#### 9.14.2.18 WARNING\_DOUBLE\_RESOURCE\_ID

```
#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 is\_error()

```
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 115 of file StatusCodes.hxx.

**9.14.3.2 is\_warning()**

```
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 104 of file StatusCodes.hxx.

**9.14.3.3 no\_error()**

```
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 81 of file StatusCodes.hxx.

**9.14.3.4 success()**

```
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**

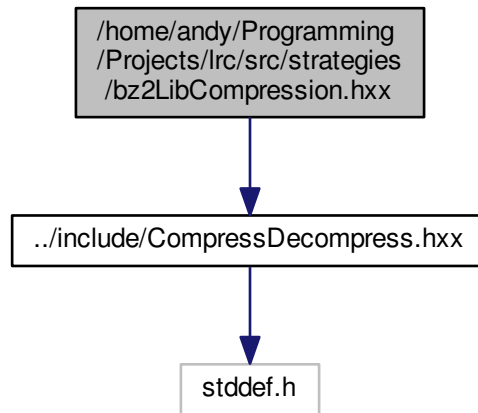
<i>true</i>	Value means success
<i>false</i>	Value means warning or error

Definition at line 92 of file StatusCodes.hxx.

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

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

Include dependency graph for bz2LibCompression.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 Tschärner

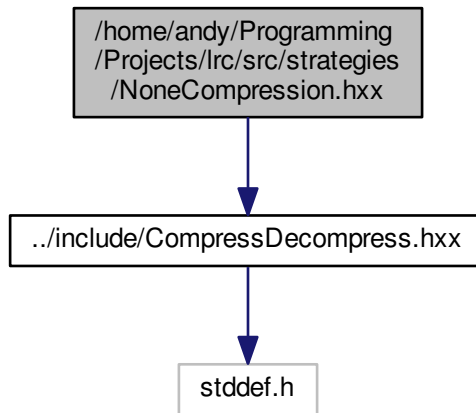
#### Date

2012-01-18

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

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

Include dependency graph for NoneCompression.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 Tschärner

#### Date

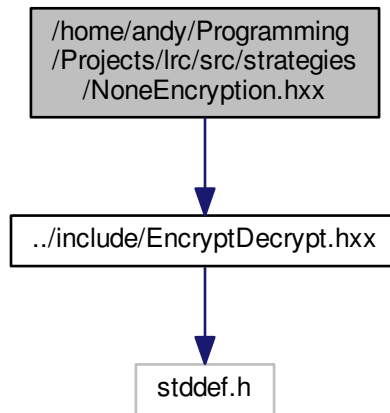
2011-06-25

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

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



Include dependency graph for NoneEncryption.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 Tschärner

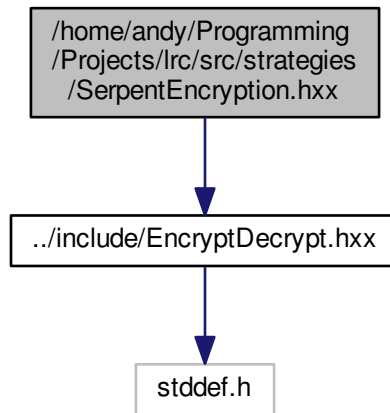
#### Date

0211-07-21

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

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

Include dependency graph for SerpentEncryption.hxx:



## Classes

- class [SerpentEncryption](#)

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

### 9.18.1 Detailed Description

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

#### Author

Andreas Tschärner

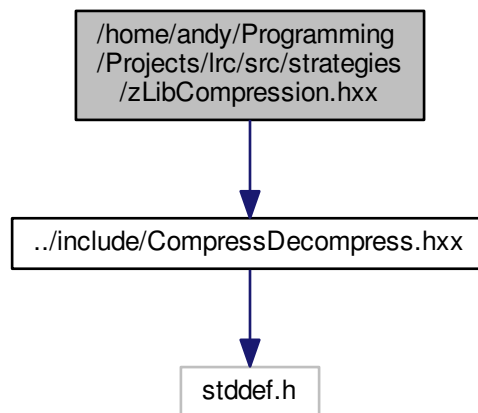
#### Date

2014-08-23

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

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

Include dependency graph for zLibCompression.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

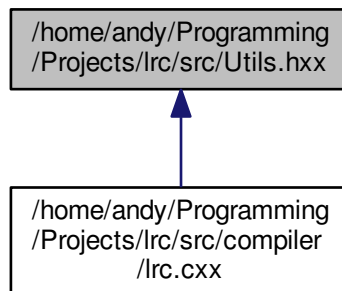
Andreas Tschärner

## Date

2011-08-14

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

This graph shows which files directly or indirectly include this file:



### Macros

- `#define DEBUG\_PRINT\(x\)`  
*Macro for debugging.*

### Functions

- `bool file\_exists (const char *p_filename)`  
*Check if a file exists.*
- `int file\_size (const char *p_filename)`  
*Returns the size of a file.*
- `char * get\_extension (char *p_filename)`  
*Returns the file extension.*
- `char * replace\_extension (char *p_filename, const char *p_newExt)`  
*Replace the file extension.*
- `void delete\_list (unsigned char **p_dataList, unsigned int p_dataSize)`  
*Delete list of data.*
- `int password\_len (const unsigned char *p_password)`  
*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 Tschärner

### Date

2014-08-23

## 9.20.2 Macro Definition Documentation

### 9.20.2.1 DEBUG\_PRINT

```
#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 delete\_list()

```
void delete_list (  
    unsigned char ** p_dataList,  
    unsigned int p_dataSize )
```

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_dataSize</i>	Size of data list

### 9.20.3.2 file\_exists()

```
bool file_exists (
    const char * p_filename )
```

Check if a file exists.

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

#### Parameters

in	<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 file\_size()

```
int file_size (
    const char * p_filename )
```

Returns the size of a file.

This function returns the size of a file

#### Parameters

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

#### Returns

Size of file

#### Remarks

If an error occurs, the function returns -1

### 9.20.3.4 get\_extension()

```
char* get_extension (
    char * p_filename )
```

Returns the file extension.

This function returns the file extension of the given file

**Parameters**

in	<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 password\_len()**

```
int password_len (
    const unsigned char * p_password )
```

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**

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

**Returns**

Length of zero-terminated password

**Remarks**

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

**9.20.3.6 replace\_extension()**

```
char* replace_extension (
    char * p_filename,
    const char * p_newExt )
```

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**

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

**Returns**

Filename with new extension

**Remarks**

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



# Index

- /home/andy/Programming/Projects/lrc/src/Factories.hxx, 90
- /home/andy/Programming/Projects/lrc/src/ResourceData.hxx, 98
- /home/andy/Programming/Projects/lrc/src/StatusCodes.hxx, 99
- /home/andy/Programming/Projects/lrc/src/Utils.hxx, 112
- /home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx, 81
- /home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx, 82
- /home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx, 86
- /home/andy/Programming/Projects/lrc/src/compiler/RCParse.hxx, 87
- /home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx, 89
- /home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx, 84
- /home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx, 91
- /home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx, 92
- /home/andy/Programming/Projects/lrc/src/include/Resource.hxx, 93
- /home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx, 96
- /home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx, 97
- /home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.hxx, 107
- /home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx, 108
- /home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx, 109
- /home/andy/Programming/Projects/lrc/src/strategies/bz2LibCompression.hxx, 106
- /home/andy/Programming/Projects/lrc/src/strategies/zLibCompression.hxx, 110
- ~Collector
  - Collector, 18
- ~InFileParser
  - InFileParser, 30
- ~Resource
  - lrc::Resource, 53
- ~ResourceData
  - ResourceData, 58
- ~ResourceManager
  - lrc::ResourceManager, 67
- ~lrcEncryptionDisabledException
  - lrcEncryptionDisabledException, 36
- ~lrcFileExistsException
  - lrcFileExistsException, 39
- ~lrcFileNotFoundException
  - lrcFileNotFoundException, 41
- are\_resIDs\_unique
  - Collector, 18
- bz2LibCompression, 15
  - compress, 16
  - decompress, 16
- clear\_internal\_error
  - InFileParser, 31
- collect
  - Collector, 19
- Collector, 17
  - ~Collector, 18
  - are\_resIDs\_unique, 18
  - collect, 19
  - Collector, 17
  - m\_rcName, 20
  - m\_rdfName, 20
  - show\_resource\_data\_error, 19
- compType
  - resEntry\_, 51
- compress
  - bz2LibCompression, 16
  - lrc::CompressDecompress, 21
  - NoneCompression, 43
  - zLibCompression, 79
- CompressionFactory, 23
  - get\_compression\_class, 23
- CompressionType
  - lrc, 14
- convert\_to\_elf
  - lrc.cxx, 85
- copy\_mandatory\_data
  - RCParse, 49
- create\_initialization\_vector
  - SerpentEncryption, 76
- create\_input\_parser
  - ParserFactory, 46
- DEBUG\_PRINT
  - Utils.hxx, 113
- decompress
  - bz2LibCompression, 16

- lrc::CompressDecompress, [22](#)
- NoneCompression, [43](#)
- zLibCompression, [79](#)
- decompress\_data
  - lrc::ResourceManager, [67](#)
- decrypt
  - lrc::EncryptDecrypt, [24](#)
  - NoneEncryption, [45](#)
  - SerpentEncryption, [77](#)
- decrypt\_data
  - lrc::ResourceManager, [68](#)
- delete\_list
  - Utils.hxx, [113](#)
- ERROR\_BASE
  - StatusCodes.hxx, [101](#)
- ERROR\_COMPRESSION\_COMPRESS
  - StatusCodes.hxx, [101](#)
- ERROR\_COMPRESSION\_DECOMPRESS
  - StatusCodes.hxx, [101](#)
- ERROR\_COMPRESSION\_NOT\_AVAILABLE
  - StatusCodes.hxx, [101](#)
- ERROR\_ENCRYPTION\_DECRYPT
  - StatusCodes.hxx, [102](#)
- ERROR\_ENCRYPTION\_ENCRYPT
  - StatusCodes.hxx, [102](#)
- ERROR\_ENCRYPTION\_NOT\_AVAILABLE
  - StatusCodes.hxx, [102](#)
- ERROR\_FILE\_NOT\_FOUND
  - StatusCodes.hxx, [102](#)
- ERROR\_FILE\_OPEN
  - StatusCodes.hxx, [102](#)
- ERROR\_FILE\_READ
  - StatusCodes.hxx, [103](#)
- ERROR\_FILE\_WRITE
  - StatusCodes.hxx, [103](#)
- ERROR\_INVALID\_PARAMETER
  - StatusCodes.hxx, [103](#)
- ERROR\_NOT\_ENOUGH\_MEMORY
  - StatusCodes.hxx, [103](#)
- ERROR\_PARSE
  - StatusCodes.hxx, [103](#)
- ERROR\_UNKNOWN\_ARCH
  - StatusCodes.hxx, [104](#)
- encType
  - resEntry\_, [51](#)
- encrypt
  - lrc::EncryptDecrypt, [25](#)
  - NoneEncryption, [45](#)
  - SerpentEncryption, [77](#)
- EncryptionFactory, [26](#)
  - get\_encryption\_class, [26](#)
- EncryptionType
  - lrc, [14](#)
- eval\_compression\_type
  - InFileParser, [31](#)
- eval\_encryption\_type
  - InFileParser, [31](#)
- file\_exists
  - Utils.hxx, [113](#)
- file\_size
  - Utils.hxx, [114](#)
- get\_ID
  - lrc::Resource, [54](#)
- get\_compression
  - ResourceData, [58](#)
- get\_compression\_class
  - CompressionFactory, [23](#)
- get\_data\_from\_memory
  - ResourceData, [58](#)
- get\_encryption
  - ResourceData, [59](#)
- get\_encryption\_class
  - EncryptionFactory, [26](#)
- get\_error\_msg
  - ResourceData, [59](#)
- get\_extension
  - Utils.hxx, [114](#)
- get\_file
  - ResourceData, [59](#)
- get\_internal\_error
  - InFileParser, [32](#)
- get\_password
  - InFileParser, [32](#)
- get\_rc\_position
  - ResourceData, [60](#)
- get\_res\_data
  - lrc::Resource, [54](#)
- get\_res\_size
  - lrc::Resource, [54](#)
- get\_resource
  - lrc::ResourceManager, [68, 69](#)
- get\_resource\_entries
  - InFileParser, [33](#)
- get\_resource\_ids
  - lrc::ResourceManager, [69](#)
- InFileParser, [27](#)
  - ~InFileParser, [30](#)
  - clear\_internal\_error, [31](#)
  - eval\_compression\_type, [31](#)
  - eval\_encryption\_type, [31](#)
  - get\_internal\_error, [32](#)
  - get\_password, [32](#)
  - get\_resource\_entries, [33](#)
  - InFileParser, [30](#)
  - internalErrorType, [29](#)
  - m\_errorPosition, [34](#)
  - m\_filename, [34](#)
  - m\_internalError, [34](#)
  - m\_lastError, [34](#)
  - m\_resEntries, [34](#)
  - parse, [33](#)
- inFilePosition
  - ResourceData.hxx, [99](#)
- internalErrorType

- InFileParser, 29
- is\_comment
  - RCParse, 49
- is\_error
  - StatusCodes.hxx, 104
- is\_warning
  - StatusCodes.hxx, 105
- is\_windows\_line
  - RCParse, 50
- load\_embedded
  - Irc::ResourceManager, 70
- load\_from\_file
  - Irc::ResourceManager, 70
- Irc, 13
  - CompressionType, 14
  - EncryptionType, 14
- Irc.cxx
  - convert\_to\_elf, 85
  - main, 85
  - usage, 85
  - VERSION, 85
- Irc::CompressDecompress, 21
  - compress, 21
  - decompress, 22
- Irc::EncryptDecrypt, 24
  - decrypt, 24
  - encrypt, 25
- Irc::Resource, 52
  - ~Resource, 53
  - get\_ID, 54
  - get\_res\_data, 54
  - get\_res\_size, 54
  - m\_resData, 55
  - m\_resID, 55
  - m\_resSize, 55
  - Resource, 53
- Irc::ResourceManager, 64
  - ~ResourceManager, 67
  - decompress\_data, 67
  - decrypt\_data, 68
  - get\_resource, 68, 69
  - get\_resource\_ids, 69
  - load\_embedded, 70
  - load\_from\_file, 70
  - m\_compType, 71
  - m\_encType, 71
  - m\_numResEntries, 71
  - m\_password, 71
  - m\_resData, 72
  - m\_resDataSize, 72
  - m\_resEntries, 72
  - m\_resourceFile, 72
  - ResourceManager, 66
  - setup\_resources, 70
- IrcEncryptionDisabledException, 35
  - ~IrcEncryptionDisabledException, 36
  - IrcEncryptionDisabledException, 36
  - m\_resourceID, 37
  - what, 37
- IrcFileExistsException, 38
  - ~IrcFileExistsException, 39
  - IrcFileExistsException, 39
  - m\_fileOverwrite, 40
  - what, 39
- IrcFileNotFoundException, 40
  - ~IrcFileNotFoundException, 41
  - IrcFileNotFoundException, 41
  - m\_fileNotFound, 42
  - what, 42
- m\_compType
  - Irc::ResourceManager, 71
- m\_compression
  - ResourceData, 63
- m\_encType
  - Irc::ResourceManager, 71
- m\_encryption
  - ResourceData, 63
- m\_errorMsg
  - ResourceData, 63
- m\_errorPosition
  - InFileParser, 34
- m\_fileNotFound
  - IrcFileNotFoundException, 42
- m\_fileOverwrite
  - IrcFileExistsException, 40
- m\_filename
  - InFileParser, 34
  - ResourceData, 63
- m\_inFilePosition
  - ResourceData, 64
- m\_internalError
  - InFileParser, 34
- m\_lastError
  - InFileParser, 34
- m\_numResEntries
  - Irc::ResourceManager, 71
- m\_password
  - Irc::ResourceManager, 71
  - ResourceData, 64
- m\_rcName
  - Collector, 20
- m\_rdfName
  - Collector, 20
- m\_resData
  - Irc::Resource, 55
  - Irc::ResourceManager, 72
- m\_resDataSize
  - Irc::ResourceManager, 72
- m\_resEntries
  - InFileParser, 34
  - Irc::ResourceManager, 72
- m\_resID
  - Irc::Resource, 55
- m\_resSize
  - Irc::Resource, 55
- m\_resourceFile

- Irc::ResourceManager, 72
- m\_resourceID
  - IrcEncryptionDisabledException, 37
- MAX\_ID\_LEN
  - Resource.hxx, 95
- main
  - Irc.cxx, 85
- NO\_ERROR
  - StatusCodes.hxx, 104
- no\_error
  - StatusCodes.hxx, 105
- NoneCompression, 42
  - compress, 43
  - decompress, 43
- NoneEncryption, 44
  - decrypt, 45
  - encrypt, 45
- parse
  - InFileParser, 33
  - RCParse, 50
  - RIFParser, 74
- ParserFactory, 46
  - create\_input\_parser, 46
- password\_len
  - Utils.hxx, 115
- prepare\_resource\_from\_file
  - ResourceData, 60
- RCParse, 47
  - copy\_mandatory\_data, 49
  - is\_comment, 49
  - is\_windows\_line, 50
  - parse, 50
  - RCParse, 48
- RIFParser, 73
  - parse, 74
  - RIFParser, 74
- replace\_extension
  - Utils.hxx, 115
- resEntry
  - Resource.hxx, 95
- resEntry\_, 51
  - compType, 51
  - encType, 51
  - resID, 51
  - resSize, 52
  - startOffset, 52
- resID
  - resEntry\_, 51
- resSize
  - resEntry\_, 52
- Resource
  - Irc::Resource, 53
- Resource.hxx
  - MAX\_ID\_LEN, 95
  - resEntry, 95
- ResourceData, 56
  - ~ResourceData, 58
  - get\_compression, 58
  - get\_data\_from\_memory, 58
  - get\_encryption, 59
  - get\_error\_msg, 59
  - get\_file, 59
  - get\_rc\_position, 60
  - m\_compression, 63
  - m\_encryption, 63
  - m\_errorMsg, 63
  - m\_filename, 63
  - m\_inFilePosition, 64
  - m\_password, 64
  - prepare\_resource\_from\_file, 60
  - ResourceData, 58
  - set\_compression, 61
  - set\_encryption, 61
  - set\_error\_msg, 61
  - set\_file, 62
  - set\_ident, 62
  - set\_rc\_position, 62
- ResourceData.hxx
  - inFilePosition, 99
- ResourceManager
  - Irc::ResourceManager, 66
- SerpentEncryption, 75
  - create\_initialization\_vector, 76
  - decrypt, 77
  - encrypt, 77
- set\_compression
  - ResourceData, 61
- set\_encryption
  - ResourceData, 61
- set\_error\_msg
  - ResourceData, 61
- set\_file
  - ResourceData, 62
- set\_ident
  - ResourceData, 62
- set\_rc\_position
  - ResourceData, 62
- setup\_resources
  - Irc::ResourceManager, 70
- show\_resource\_data\_error
  - Collector, 19
- startOffset
  - resEntry\_, 52
- StatusCodes.hxx
  - ERROR\_BASE, 101
  - ERROR\_COMPRESSION\_COMPRESS, 101
  - ERROR\_COMPRESSION\_DECOMPRESS, 101
  - ERROR\_COMPRESSION\_NOT\_AVAILABLE, 101
  - ERROR\_ENCRYPTION\_DECRYPT, 102
  - ERROR\_ENCRYPTION\_ENCRYPT, 102
  - ERROR\_ENCRYPTION\_NOT\_AVAILABLE, 102
  - ERROR\_FILE\_NOT\_FOUND, 102
  - ERROR\_FILE\_OPEN, 102
  - ERROR\_FILE\_READ, 103

- ERROR\_FILE\_WRITE, [103](#)
- ERROR\_INVALID\_PARAMETER, [103](#)
- ERROR\_NOT\_ENOUGH\_MEMORY, [103](#)
- ERROR\_PARSE, [103](#)
- ERROR\_UNKNOWN\_ARCH, [104](#)
- is\_error, [104](#)
- is\_warning, [105](#)
- NO\_ERROR, [104](#)
- no\_error, [105](#)
- success, [106](#)
- WARNING\_BASE, [104](#)
- WARNING\_DOUBLE\_RESOURCE\_ID, [104](#)

success

- StatusCodes.hxx, [106](#)

usage

- Irc.cxx, [85](#)

Utils.hxx

- DEBUG\_PRINT, [113](#)
- delete\_list, [113](#)
- file\_exists, [113](#)
- file\_size, [114](#)
- get\_extension, [114](#)
- password\_len, [115](#)
- replace\_extension, [115](#)

VERSION

- Irc.cxx, [85](#)

WARNING\_BASE

- StatusCodes.hxx, [104](#)

WARNING\_DOUBLE\_RESOURCE\_ID

- StatusCodes.hxx, [104](#)

what

- IrcEncryptionDisabledException, [37](#)
- IrcFileExistsException, [39](#)
- IrcFileNotFoundException, [42](#)

zLibCompression, [78](#)

- compress, [79](#)
- decompress, [79](#)