## Contents

1. **Irc - Linux Resource Compiler**  
   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
8.5.2 decrypt() ................................................. 24
8.5.2 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 \sim{} 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 lrcEncryptionDisabledException Class Reference ......................... 35
8.8.1 Detailed Description ................................................ 36
8.8.2 Constructor & Destructor Documentation ............................... 36
8.8.2.1 lrcEncryptionDisabledException() .................................................. 36
8.8.2.2 ~lrcEncryptionDisabledException() ................................................... 36
8.8.3 Member Function Documentation ......................................................... 37
  8.8.3.1 what() ................................................................................. 37
8.8.4 Member Data Documentation ............................................................... 37
  8.8.4.1 m_resourceID ........................................................................ 37
8.9 lrcFileExistsException Class Reference .................................................. 38
  8.9.1 Detailed Description ....................................................................... 39
  8.9.2 Constructor & Destructor Documentation ....................................... 39
    8.9.2.1 lrcFileExistsException() ......................................................... 39
    8.9.2.2 ~lrcFileExistsException() ......................................................... 39
  8.9.3 Member Function Documentation ..................................................... 39
    8.9.3.1 what() ................................................................................. 39
  8.9.4 Member Data Documentation .............................................................. 40
    8.9.4.1 m_fileOverwrite ..................................................................... 40
8.10 lrcFileNotFoundException Class Reference .............................................. 40
  8.10.1 Detailed Description ................................................................... 41
  8.10.2 Constructor & Destructor Documentation .................................... 41
    8.10.2.1 lrcFileNotFoundException() .................................................... 41
    8.10.2.2 ~lrcFileNotFoundException() .................................................... 41
  8.10.3 Member Function Documentation .................................................. 42
    8.10.3.1 what() ................................................................................. 42
  8.10.4 Member Data Documentation .......................................................... 42
    8.10.4.1 m_fileNotFound ................................................................... 42
8.11 NoneCompression Class Reference ..................................................... 42
  8.11.1 Detailed Description ................................................................... 43
  8.11.2 Member Function Documentation ................................................. 43
    8.11.2.1 compress() ......................................................................... 43
    8.11.2.2 decompress() ..................................................................... 44
8.12 NoneEncryption Class Reference ............................................................. 44
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.12.1</td>
<td>Detailed Description</td>
<td>45</td>
</tr>
<tr>
<td>8.12.2</td>
<td>Member Function Documentation</td>
<td>45</td>
</tr>
<tr>
<td>8.12.2.1</td>
<td>decrypt()</td>
<td>45</td>
</tr>
<tr>
<td>8.12.2.2</td>
<td>encrypt()</td>
<td>45</td>
</tr>
<tr>
<td>8.13</td>
<td>ParserFactory Class Reference</td>
<td>46</td>
</tr>
<tr>
<td>8.13.1</td>
<td>Detailed Description</td>
<td>46</td>
</tr>
<tr>
<td>8.13.2</td>
<td>Member Function Documentation</td>
<td>46</td>
</tr>
<tr>
<td>8.13.2.1</td>
<td>create_input_parser()</td>
<td>46</td>
</tr>
<tr>
<td>8.14</td>
<td>RCParser Class Reference</td>
<td>47</td>
</tr>
<tr>
<td>8.14.1</td>
<td>Detailed Description</td>
<td>48</td>
</tr>
<tr>
<td>8.14.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>48</td>
</tr>
<tr>
<td>8.14.2.1</td>
<td>RCParser()</td>
<td>48</td>
</tr>
<tr>
<td>8.14.3</td>
<td>Member Function Documentation</td>
<td>49</td>
</tr>
<tr>
<td>8.14.3.1</td>
<td>copy_mandatory_data()</td>
<td>49</td>
</tr>
<tr>
<td>8.14.3.2</td>
<td>is_comment()</td>
<td>49</td>
</tr>
<tr>
<td>8.14.3.3</td>
<td>is_windows_line()</td>
<td>50</td>
</tr>
<tr>
<td>8.14.3.4</td>
<td>parse()</td>
<td>50</td>
</tr>
<tr>
<td>8.15</td>
<td>resEntry_ Struct Reference</td>
<td>51</td>
</tr>
<tr>
<td>8.15.1</td>
<td>Detailed Description</td>
<td>51</td>
</tr>
<tr>
<td>8.15.2</td>
<td>Member Data Documentation</td>
<td>51</td>
</tr>
<tr>
<td>8.15.2.1</td>
<td>compType</td>
<td>51</td>
</tr>
<tr>
<td>8.15.2.2</td>
<td>encType</td>
<td>51</td>
</tr>
<tr>
<td>8.15.2.3</td>
<td>resID</td>
<td>52</td>
</tr>
<tr>
<td>8.15.2.4</td>
<td>resSize</td>
<td>52</td>
</tr>
<tr>
<td>8.15.2.5</td>
<td>startOffset</td>
<td>52</td>
</tr>
<tr>
<td>8.16</td>
<td>lrc::Resource Class Reference</td>
<td>52</td>
</tr>
<tr>
<td>8.16.1</td>
<td>Detailed Description</td>
<td>53</td>
</tr>
<tr>
<td>8.16.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>53</td>
</tr>
<tr>
<td>8.16.2.1</td>
<td>Resource()</td>
<td>53</td>
</tr>
<tr>
<td>8.16.2.2</td>
<td>~Resource()</td>
<td>54</td>
</tr>
</tbody>
</table>
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

Generated by Doxygen
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.19.2</td>
<td>Constructor &amp; Destructor Documentation</td>
<td>74</td>
</tr>
<tr>
<td>8.19.2.1</td>
<td>RIFParser()</td>
<td>74</td>
</tr>
<tr>
<td>8.19.3</td>
<td>Member Function Documentation</td>
<td>74</td>
</tr>
<tr>
<td>8.19.3.1</td>
<td>parse()</td>
<td>74</td>
</tr>
<tr>
<td>8.20</td>
<td>SerpentEncryption Class Reference</td>
<td>75</td>
</tr>
<tr>
<td>8.20.1</td>
<td>Detailed Description</td>
<td>76</td>
</tr>
<tr>
<td>8.20.2</td>
<td>Member Function Documentation</td>
<td>76</td>
</tr>
<tr>
<td>8.20.2.1</td>
<td>create_initialization_vector()</td>
<td>76</td>
</tr>
<tr>
<td>8.20.2.2</td>
<td>decrypt()</td>
<td>77</td>
</tr>
<tr>
<td>8.20.2.3</td>
<td>encrypt()</td>
<td>77</td>
</tr>
<tr>
<td>8.21</td>
<td>zLibCompression Class Reference</td>
<td>78</td>
</tr>
<tr>
<td>8.21.1</td>
<td>Detailed Description</td>
<td>78</td>
</tr>
<tr>
<td>8.21.2</td>
<td>Member Function Documentation</td>
<td>79</td>
</tr>
<tr>
<td>8.21.2.1</td>
<td>compress()</td>
<td>79</td>
</tr>
<tr>
<td>8.21.2.2</td>
<td>decompress()</td>
<td>79</td>
</tr>
<tr>
<td>9</td>
<td>File Documentation</td>
<td>81</td>
</tr>
<tr>
<td>9.1</td>
<td>/home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx File Reference</td>
<td>81</td>
</tr>
<tr>
<td>9.1.1</td>
<td>Detailed Description</td>
<td>82</td>
</tr>
<tr>
<td>9.2</td>
<td>/home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx File Reference</td>
<td>82</td>
</tr>
<tr>
<td>9.2.1</td>
<td>Detailed Description</td>
<td>83</td>
</tr>
<tr>
<td>9.3</td>
<td>/home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx File Reference</td>
<td>84</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Detailed Description</td>
<td>85</td>
</tr>
<tr>
<td>9.3.2</td>
<td>Macro Definition Documentation</td>
<td>85</td>
</tr>
<tr>
<td>9.3.2.1</td>
<td>VERSION</td>
<td>85</td>
</tr>
<tr>
<td>9.3.3</td>
<td>Function Documentation</td>
<td>85</td>
</tr>
<tr>
<td>9.3.3.1</td>
<td>convert_to_elf()</td>
<td>85</td>
</tr>
<tr>
<td>9.3.3.2</td>
<td>main()</td>
<td>85</td>
</tr>
<tr>
<td>9.3.3.3</td>
<td>usage()</td>
<td>86</td>
</tr>
<tr>
<td>9.4</td>
<td>/home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx File Reference</td>
<td>86</td>
</tr>
<tr>
<td>9.4.1</td>
<td>Detailed Description</td>
<td>87</td>
</tr>
<tr>
<td>Section</td>
<td>File Path</td>
<td>Reference Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>9.5</td>
<td>/home/andy/Programming/Projects/lrc/src/compiler/RCParser.hxx</td>
<td>87</td>
</tr>
<tr>
<td>9.5.1</td>
<td>Detailed Description</td>
<td>88</td>
</tr>
<tr>
<td>9.6</td>
<td>/home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx</td>
<td>89</td>
</tr>
<tr>
<td>9.6.1</td>
<td>Detailed Description</td>
<td>89</td>
</tr>
<tr>
<td>9.7</td>
<td>/home/andy/Programming/Projects/lrc/src/Factories.hxx</td>
<td>90</td>
</tr>
<tr>
<td>9.7.1</td>
<td>Detailed Description</td>
<td>90</td>
</tr>
<tr>
<td>9.8</td>
<td>/home/andy/Programming/Projects/lrc/src/include/CompressDecompress.hxx</td>
<td>91</td>
</tr>
<tr>
<td>9.8.1</td>
<td>Detailed Description</td>
<td>92</td>
</tr>
<tr>
<td>9.9</td>
<td>/home/andy/Programming/Projects/lrc/src/include/EncryptDecrypt.hxx</td>
<td>92</td>
</tr>
<tr>
<td>9.9.1</td>
<td>Detailed Description</td>
<td>93</td>
</tr>
<tr>
<td>9.10</td>
<td>/home/andy/Programming/Projects/lrc/src/include/Resource.hxx</td>
<td>93</td>
</tr>
<tr>
<td>9.10.1</td>
<td>Detailed Description</td>
<td>95</td>
</tr>
<tr>
<td>9.10.2</td>
<td>Macro Definition Documentation</td>
<td>95</td>
</tr>
<tr>
<td>9.10.2.1</td>
<td>MAX_ID_LEN</td>
<td>95</td>
</tr>
<tr>
<td>9.10.3</td>
<td>Typedef Documentation</td>
<td>95</td>
</tr>
<tr>
<td>9.10.3.1</td>
<td>resEntry</td>
<td>95</td>
</tr>
<tr>
<td>9.11</td>
<td>/home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx</td>
<td>96</td>
</tr>
<tr>
<td>9.11.1</td>
<td>Detailed Description</td>
<td>96</td>
</tr>
<tr>
<td>9.12</td>
<td>/home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx</td>
<td>97</td>
</tr>
<tr>
<td>9.12.1</td>
<td>Detailed Description</td>
<td>97</td>
</tr>
<tr>
<td>9.13</td>
<td>/home/andy/Programming/Projects/lrc/src/ResourceData.hxx</td>
<td>98</td>
</tr>
<tr>
<td>9.13.1</td>
<td>Detailed Description</td>
<td>99</td>
</tr>
<tr>
<td>9.13.2</td>
<td>Typedef Documentation</td>
<td>99</td>
</tr>
<tr>
<td>9.13.2.1</td>
<td>inFilePosition</td>
<td>99</td>
</tr>
<tr>
<td>9.14</td>
<td>/home/andy/Programming/Projects/lrc/src/StatusCodes.hxx</td>
<td>99</td>
</tr>
<tr>
<td>9.14.1</td>
<td>Detailed Description</td>
<td>101</td>
</tr>
<tr>
<td>9.14.2</td>
<td>Macro Definition Documentation</td>
<td>101</td>
</tr>
<tr>
<td>9.14.2.1</td>
<td>ERROR_BASE</td>
<td>101</td>
</tr>
<tr>
<td>9.14.2.2</td>
<td>ERROR_COMPRESSION_COMPRESS</td>
<td>101</td>
</tr>
<tr>
<td>9.14.2.3</td>
<td>ERROR_COMPRESSION_DECOMPRESS</td>
<td>101</td>
</tr>
</tbody>
</table>
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>|<rif-File>
```

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 lblrc can be found in the src/example directory.

Author

Andreas Tscharner

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:

- **lrc**
  
  Namespace lrc for classes in the library and for extending lrc
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 ............................................................................................................ 35
   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:

- **bz2LibCompression**: Compression class that uses the bzip2 algorithm ........... 15
- **Collector**: Class to collect all resource data ........................................... 17
- **lrc::CompressDecompress**: Compression and Decompression base class ........ 21
- **CompressionFactory**: Factory to create compression class instances .......... 23
- **lrc::EncryptDecrypt**: Encryption and Decryption base class .................... 24
- **EncryptionFactory**: Factory to create encryption class instances ............... 26
- **InFileParser**: Base class for all .rc input files .................................... 27
- **lrcEncryptionDisabledException**: Exception if encryption is disabled but required ... 35
- **lrcFileExistsException**: Exception if an existing file should be overwritten .... 38
- **lrcFileNotFoundException**: Exception class if a file could not be found ........ 40
- **NoneCompression**: Compression class that does NO compression ............... 42
- **NoneEncryption**: Encryption class that does NO encryption .................... 44
- **ParserFactory**: Factory class to create an appropriate parser class .......... 46
- **RCParser**: Class to parse an .rc file .................................................... 47
- **resEntry_**: Data for one resource entry ................................................. 51
- **lrc::Resource**: Resource class for use with library ................................ 52
- **ResourceData**: Internal class for resource with more information .............. 56
- **lrc::ResourceManager**: Manager class handling all resources of a resource file ... 64
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIFParser</td>
<td>Class to parse a .rif (XML) file</td>
<td>73</td>
</tr>
<tr>
<td>SerpentEncryption</td>
<td>Class to encrypt/decrypt using the Serpent algorithm</td>
<td>75</td>
</tr>
<tr>
<td>zLibCompression</td>
<td>Compression class that uses the zLib algorithm</td>
<td>78</td>
</tr>
</tbody>
</table>
Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

/home/andy/Programming/Projects/lrc/src/Factories.hxx .................................................. 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/RCParser.hxx .......................................... 84
/home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx .......................................... 86
/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/ResourceManager.hxx ................................ 96
/home/andy/Programming/Projects/lrc/src/strategies/b22LibCompression.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 lrc Namespace Reference

Namespace lrc for classes in the library and for extending lrc.

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 lrc for classes in the library and for extending lrc.

The namespace lrc was introduced and used for classes and functions that will be used in the librclrc 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

```cpp
enum lrc::CompressionType
```

Possible compression types for resource

**Todo**  Add more possibilities to compress resource

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoneCompression</td>
<td>No compression at all.</td>
</tr>
<tr>
<td>zLibCompression</td>
<td>zlib compression</td>
</tr>
<tr>
<td>bz2LibCompression</td>
<td>bzip2 compression</td>
</tr>
<tr>
<td>lastCompression</td>
<td>Marker for last entry.</td>
</tr>
</tbody>
</table>

Definition at line 44 of file CompressDecompress.hxx.

7.1.2.2  EncryptionType

```cpp
enum lrc::EncryptionType
```

Possible encryption types for resource

**Todo**  Add more possibilities to encrypt resource

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoneEncryption</td>
<td>No encryption at all.</td>
</tr>
<tr>
<td>SerpentEncryption</td>
<td>Serpent algorithm for encryption.</td>
</tr>
<tr>
<td>lastEncryption</td>
<td>Marker for last entry.</td>
</tr>
</tbody>
</table>

Definition at line 44 of file EncryptDecrypt.hxx.
Chapter 8

Class Documentation

8.1  bz2LibCompression Class Reference

Compression class that uses the *bz*p2 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 lrc::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 lrc::CompressDecompress.

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

- /home/andy/Programming/Projects/lrc/src estratégias/bz2LibCompression.hxx

Generated by Doxygen
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 (lrcFileExistsException)
  Constructor.
- **~Collector (void)**
  Destructor.
- **int collect (std::vector<ResourceData ∗> ∗p_resEntries, lrc::CompressionType p_compress, lrc::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 (lrcFileExistsException)

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.
8.2.2.2 \texttt{\~{}Collector()}


\texttt{Collector::\~{}Collector (}
  \texttt{void )}

Destructor.

Clean up the memory that was needed by the class.

8.2.3 Member Function Documentation

8.2.3.1 \texttt{are\_resIDs\_unique()}

\begin{verbatim}
int Collector::are_resIDs_unique ( 
  std::vector<ResourceData *> p_entries, 
  inFilePosition & p_doubleIDPos, 
  char ** p_doubleID ) [private]
\end{verbatim}

Check for unique resource IDs.

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

Parameters

\begin{tabular}{|c|c|}
\hline
\textbf{in} & \textit{p\_entries} & List of resource data (of type resEntry) \\
\hline
\textbf{out} & \textit{p\_doubleIDPos} & Position of double ID \\
\hline
\textbf{out} & \textit{p\_doubleID} & Resource ID that is not unique \\
\hline
\end{tabular}

Return values

\begin{tabular}{|c|c|}
\hline
\textit{NO\_ERROR} & All IDs are unique \\
\hline
\textit{WARNING\_DOUBLE\_RESOURCE\_ID} & A resource ID appears more than once \\
\hline
\end{tabular}
Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_INVALID_PARAMETER</td>
<td>One or more parameters are invalid</td>
</tr>
</tbody>
</table>

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()

```cpp
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

<table>
<thead>
<tr>
<th>in</th>
<th>p_resEntries</th>
<th>Collection of data resource entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_compress</td>
<td>Compression type for complete file</td>
</tr>
<tr>
<td>in</td>
<td>p_encrypt</td>
<td>Encryption type for complete file</td>
</tr>
<tr>
<td>in</td>
<td>p_key</td>
<td>Password if encryption is requested</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>Resource Data File successfully compiled</td>
</tr>
<tr>
<td>ERROR_FILE_OPEN</td>
<td>An error occurred opening the rdf file</td>
</tr>
<tr>
<td>ERROR_FILE_NOT_FOUND</td>
<td>The desired resource data file could not be found</td>
</tr>
<tr>
<td>ERROR_INVALID_PARAMETER</td>
<td>The provided parameter was nullptr</td>
</tr>
<tr>
<td>ERROR_FILE_NOT_FOUND</td>
<td>The resource file could not be found</td>
</tr>
<tr>
<td>ERROR_COMPRESSION_NOT_AVAILABLE</td>
<td>The selected compression is not available</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_NOT_AVAILABLE</td>
<td>An error occurred while encrypting the complete file</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_ENCRYPT</td>
<td>The selected encryption is not available</td>
</tr>
<tr>
<td>ERROR_FILE_READ</td>
<td>An error occurred while reading the file</td>
</tr>
<tr>
<td>ERROR_FILE_WRITE</td>
<td>An error occurred while writing the file</td>
</tr>
</tbody>
</table>
8.2.3.3 show_resource_data_error()

```c
void Collector::show_resource_data_error (  
im 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**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_errorCode</td>
<td>Error code</td>
</tr>
<tr>
<td>in</td>
<td>p_resData</td>
<td>ResourceData instance that caused the error</td>
</tr>
</tbody>
</table>

8.2.4 Member Data Documentation

8.2.4.1 m_rcName

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

Filename of .rc file.

Definition at line 50 of file Collector.hxx.

8.2.4.2 m_rdfName

```c
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.
Class Documentation

Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_decompData</td>
<td>Not yet compressed resource data</td>
</tr>
<tr>
<td>in</td>
<td>p_decompSize</td>
<td>Size of the not yet compressed resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_compData</td>
<td>Compressed resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_compSize</td>
<td>Size of the compressed resource data</td>
</tr>
</tbody>
</table>

Return values

- **NO_ERROR** | Data successfully compressed

Remarks

The caller is responsible to free the used memory

Implemented in NoneCompression, bz2LibCompression, and zLibCompression.

8.3.2.2 decompress()

```cpp
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

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_compData</td>
<td>Compressed resource data</td>
</tr>
<tr>
<td>in</td>
<td>p_compSize</td>
<td>Size of compressed resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_decompData</td>
<td>Decompressed resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_decompSize</td>
<td>Size of decompressed resource data</td>
</tr>
</tbody>
</table>

Return values

- **NO_ERROR** | 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 com-
presses 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  | p_compType | Compression type |

Returns

Instance of desired compression class

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

• /home/andy/Programming/Projects/lrc/src/Factories.hxx
8.5 lrc::EncryptDecrypt Class Reference

Encryption and Decryption base class.

#include <EncryptDecrypt.hxx>

Inheritance diagram for lrc::EncryptDecrypt:

![Inheritance Diagram](image)

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

<table>
<thead>
<tr>
<th>in</th>
<th>p_key</th>
<th>Key for decryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_encData</td>
<td>Encrypted resource data for decryption</td>
</tr>
<tr>
<td>in</td>
<td>p_encSize</td>
<td>Size of encrypted resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_clearData</td>
<td>Decrypted/clear resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_clearSize</td>
<td>Size of clear resource data</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>NO_ERROR</th>
<th>Resource data successfully decrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_INVALID_PARAMETER</td>
<td>The return buffer p_clearData was nullptr</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>p_key</th>
<th>Key for encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_clearData</td>
<td>Clear resource data for encryption</td>
</tr>
<tr>
<td>in</td>
<td>p_clearSize</td>
<td>Size of clear resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_encData</td>
<td>Encrypted resource data</td>
</tr>
<tr>
<td>out</td>
<td>p_encSize</td>
<td>Size of encrypted data</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>NO_ERROR</th>
<th>Resource data successfully encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_INVALID_PARAMETER</td>
<td>The return buffer p_encData was nullptr</td>
</tr>
</tbody>
</table>

Generated by Doxygen
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 (lrcEncryptionDisabledException)
  
  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

<table>
<thead>
<tr>
<th></th>
<th>p_encType</th>
<th>Encryption type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p_resID</td>
<td>ID of resource that requests encryption</td>
</tr>
</tbody>
</table>

Returns

Instance of desired encryption class

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

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

### 8.7 InFileParser Class Reference

Base class for all lrc input files.

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

Inheritance diagram for InFileParser:

![InFileParser Inheritance Diagram](image.png)
Collaboration diagram for InFileParser:

![Collaboration Diagram](image.png)

Public Member Functions

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

Static Public Member Functions

- static **lrc::CompressionType** eval_compression_type (const char *p_compStr)
  Evaluate compression type from string.
- static **lrc::EncryptionType** eval_encryption_type (const char *p_encStr)
  Evaluate encryption type from string.
- static unsigned char * **get_password** (const char *p_passwdStr) throw (lrcFileNotFoundException)
  Get password.
8.7 InFileParser Class Reference

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

<table>
<thead>
<tr>
<th>Enumerators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ieNone</td>
<td>No internal error.</td>
</tr>
<tr>
<td>ieInvalidElement</td>
<td>The root or any other element has a wrong name.</td>
</tr>
<tr>
<td>ieIdentNotFound</td>
<td>The identifier could not be found.</td>
</tr>
<tr>
<td>ieMissingPassword</td>
<td>An encryption is used, but no password is given.</td>
</tr>
<tr>
<td>ieFilenameNotFound</td>
<td>The actual resource filename is missing.</td>
</tr>
<tr>
<td>iePasswordFileNotFound</td>
<td>The password file could not be found.</td>
</tr>
<tr>
<td>ieUnknownCompression</td>
<td>The compression is incorrectly spelled or not available.</td>
</tr>
<tr>
<td>ieUnknownEncryption</td>
<td>The encryption is incorrectly spelled or not available.</td>
</tr>
</tbody>
</table>

Definition at line 54 of file InFileParser.hxx.

### 8.7.3 Constructor & Destructor Documentation

#### 8.7.3.1 InFileParser()

```cpp
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**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_filename</td>
<td>Filename of input file</td>
</tr>
</tbody>
</table>

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lrcFileNotFoundException</td>
<td>Exception that is thrown if the given file could not be found</td>
</tr>
</tbody>
</table>

#### 8.7.3.2 ~InFileParser()

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

Destructor.

Cleans up the memory and resources the parser class needed
8.7 InFileParser Class Reference

Exceptions

| InFileNotFoundException | 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()

```cpp
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()

```cpp
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 | p_compStr | Compression type as string |

Return values

<table>
<thead>
<tr>
<th>In::NoneCompression</th>
<th>No compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>In::zLibCompression</td>
<td>zlib compression</td>
</tr>
</tbody>
</table>

Remarks

- `In::NoneCompression` is the default if the given string defines no other compression type.

8.7.4.3 eval_encryption_type()

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

Evaluate encryption type from string.

This method evaluates which encryption type should be used, defined by the given string.
Parameters

| in  | p_encStr | Encryption type as string |

Return values

| lrc::NoneEncryption | No encryption |
| lrc::SerpentEncryption | Serpent encryption |

Remarks

lrc::NoneEncryption is the default if the given string defines no other encryption type

8.7.4.4 get_internal_error()

```cpp
virtual int InFileParser::get_internal_error (
    inFilePosition & p_errPos,
    char ** pErrMsg ) [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 | p_errPos | Line and column of error |
| out | pErrMsg  | Error message |

Returns

Error code of last error

8.7.4.5 get_password()

```cpp
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 | p_passwdStr | Password string |

Returns

Password directly from input file or from re-directed file

Exceptions

| lrcFileNotFoundException | 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

<table>
<thead>
<tr>
<th>Return value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>File successfully parsed</td>
</tr>
<tr>
<td>ERROR_FILE_OPEN</td>
<td>An error occurred while trying to open the file</td>
</tr>
<tr>
<td>ERROR_PARSE</td>
<td>An error occurred while trying to parse the file</td>
</tr>
</tbody>
</table>

Implemented in RCParse, 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/lrc/src/compiler/InFileParser.hxx
```

### 8.8  lrcEncryptionDisabledException Class Reference

Exception if encryption is disabled but required.

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

Inheritance diagram for lrcEncryptionDisabledException:

```
std::exception

lrcEncryptionDisabledException
```

Collaboration diagram for lrcEncryptionDisabledException:

```
std::exception

lrcEncryptionDisabledException
```
Public Member Functions

- **lrcEncryptionDisabledException** (char *p_resourceID)
  
  Constructor.

- **∼lrcEncryptionDisabledException** (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 lrcExceptions.hxx.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 lrcEncryptionDisabledException()

lrcEncryptionDisabledException::lrcEncryptionDisabledException ( char * p_resourceID )

Constructor.

The constructor expects the name of the resource that requires encryption.

Parameters

| in   | p_resourceID | Resource ID |

8.8.2.2 ∼lrcEncryptionDisabledException()

lrcEncryptionDisabledException::∼lrcEncryptionDisabledException ( void ) throw ()

Generated by Doxygen
Destructor.

Frees the memory of the exception

### 8.8.3 Member Function Documentation

#### 8.8.3.1 what()

```cpp
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

```cpp
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

Generated by Doxygen
8.9  lrcFileExistsException Class Reference

Exception if an existing file should be overwritten.

#include <lrcExceptions.hxx>

Inheritance diagram for lrcFileExistsException:

```
std::exception
```

```
lrcFileExistsException
```

Collaboration diagram for lrcFileExistsException:

```
std::exception
```

```
lrcFileExistsException
```

Public Member Functions

- `lrcFileExistsException (char * p_filename)`
  Constructor.
- `~lrcFileExistsException (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 lrcFileExistsException()

lrcFileExistsException::lrcFileExistsException ( char * p_filename )

Constructor.

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

Parameters

| in | p_filename | File of file to be overwritten |

8.9.2.2 ~lrcFileExistsException()

lrcFileExistsException::~lrcFileExistsException ( void ) throw )

Destructor.

Frees up memory needed by the class

8.9.3 Member Function Documentation

8.9.3.1 what()

virtual const char* lrcFileExistsException::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 lrcFileNotFoundException Class Reference

Exception class if a file could not be found.
#include <lrcExceptions.hxx>

Inheritance diagram for lrcFileNotFoundException:

Generated by Doxygen
Public Member Functions

• lrcFileNotFoundException (char ∗p_fileNotFoundName)
  Constructor.
• ∼lrcFileNotFoundException (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 lrcExceptions.hxx.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 lrcFileNotFoundException()

lrcFileNotFoundException::lrcFileNotFoundException {
    char ∗ p_fileNotFoundName }

Constructor.

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

Parameters

| in  | p_fileNotFoundName | Name of file that could not be found |

8.10.2.2 ∼lrcFileNotFoundException()

lrcFileNotFoundException::∼lrcFileNotFoundException {
    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:

```
           | +-- NoneCompression

           v
        Irc::CompressDecompress
```

Generated by Doxygen
Collaboration diagram for NoneCompression:

![Collaboration Diagram](image)

**Public Member Functions**

- `int compress (const unsigned char * const, size_t, unsigned char **, size_t &)`
  
  *No compression.*

- `int decompress (const unsigned char * const, 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()

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

No compression.

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

Implements `lrc::CompressDecompress`.
8.11.2.2 decompress()

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

No decompression.

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

Implements lrc::CompressDecompress.

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

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

### 8.12 NoneEncryption Class Reference

Encryption class that does **NO** encryption.

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

Inheritance diagram for NoneEncryption:

```
  Irc::EncryptDecrypt
     |
     v
NoneEncryption
```

Collaboration diagram for NoneEncryption:

```
  Irc::EncryptDecrypt
     |
     v
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()

```cpp
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 lrc::EncryptDecrypt.

8.12.2.2 encrypt()

```cpp
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 lrc::EncryptDecrypt.

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

- /home/andy/Programming/Projects/lrc/src/strategies/NoneEncryption.hxx

Generated by Doxygen
8.13 ParserFactory Class Reference

Factory class to create an appropriate parser class.

```cpp
#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 (xml) file.

Definition at line 46 of file ParserFactory.hxx.

8.13.2 Member Function Documentation

8.13.2.1 create_input_parser()

```cpp
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 | p_filename | 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`
Class to parse an .rc file.

#include <RCParser.hxx>

Inheritance diagram for RCParser:

Collaboration diagram for RCParser:
Public Member Functions

- **RCParser** (char *p_filename) throw (lrcFileNotFoundException)
  
  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 **RCParser** 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 RCParser.hxx.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 RCParser()

**RCParser::RCParser** (char * p_filename ) throw lrcFileNotFoundException)

Constructor.

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

Parameters

| in | p_filename | Filename of RC file |
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lrcFileNotFoundException</td>
<td>Exception that is thrown if the given .rc file could not be found</td>
</tr>
</tbody>
</table>

### 8.14.3 Member Function Documentation

#### 8.14.3.1 `copy_mandatory_data()`

```cpp
void RCParser::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 lrc match.

**Parameters**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td><code>p_mandatoryMatch</code></td>
<td>C++ match containing the data</td>
</tr>
<tr>
<td>in</td>
<td><code>p_winData</code></td>
<td>Flag indicating windows</td>
</tr>
<tr>
<td>out</td>
<td><code>p_resIdent</code></td>
<td>Resource identifier</td>
</tr>
<tr>
<td>out</td>
<td><code>p_resFilename</code></td>
<td>Resource filename</td>
</tr>
</tbody>
</table>

#### 8.14.3.2 `is_comment()`

```cpp
bool RCParser::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**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td><code>p_line</code></td>
<td>Line of text</td>
</tr>
</tbody>
</table>

**Return values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>Line is comment or empty</td>
</tr>
<tr>
<td>false</td>
<td>Line contains information</td>
</tr>
</tbody>
</table>
8.14.3.3  is_windows_line()

bool RCParser::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  | p_line | Line of RC file |

Return values

| true | It is a valid line for Windows RC |
| false | It would also fail on Windows |

8.14.3.4  parse()

int RCParser::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 ERRORPARSE, get_internal_error will provide more information

Return values

| NO_ERROR | File successfully parsed |
| ERROR_FILE_OPEN | An error occurred while trying to open the file |
| ERROR_PARSE | 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/RCParser.hxx

Generated by Doxygen
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  

lrc::Resource Class Reference

Resource class for use with library.

#include <Resource.hxx>

Inheritance diagram for lrc::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 lrc. 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()

```cpp
lrc::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()

```cpp
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()

```cpp
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()

```cpp
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()

```cpp
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.

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

Inheritance diagram for ResourceData:

```
Irc::Resource

ResourceData
```

Collaboration diagram for ResourceData:

```
Irc::Resource

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.

- `lrc::EncryptionType get_encryption (void)`
  Return encryption type.

void `set_compression (lrc::CompressionType p_resCompression)`
Set compression type.

- `lrc::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.

- `lrc::EncryptionType m_encryption`
  Type of encryption for this resource.

- `lrc::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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_dataStart</td>
<td>Start address of memory block</td>
</tr>
<tr>
<td>p_resEntry</td>
<td>Entry block of resource</td>
</tr>
<tr>
<td>p_password</td>
<td>Password if resource is encrypted</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>Resource data successfully extracted</td>
</tr>
<tr>
<td>ERROR_INVALID_PARAMETER</td>
<td>One or more provided parameters</td>
</tr>
<tr>
<td>ERROR_COMPRESSION_NOT_AVAILABLE</td>
<td>The required compression class is not available</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_NOT_AVAILABLE</td>
<td>The required encryption class is not available</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_DECRYPT</td>
<td>An error occurred while decrypting the data</td>
</tr>
<tr>
<td>ERROR_COMPRESSION_DECOMPRESS</td>
<td>An error occurred while decompressing the data</td>
</tr>
</tbody>
</table>

Remarks

The caller is responsible to free the returned class

8.17.3.3 get_encryption()

```
irc::EncryptionType ResourceData::get_encryption ( void )
```

Return encryption type.

Method to return the encryption type of this very resource

Returns

Encryption type of resource

8.17.3.4 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

<table>
<thead>
<tr>
<th>out</th>
<th>p_resData</th>
<th>Resource data for collector</th>
</tr>
</thead>
<tbody>
<tr>
<td>out</td>
<td>p_resSize</td>
<td>Size of resource data</td>
</tr>
</tbody>
</table>

Return values

<table>
<thead>
<tr>
<th>NO_ERROR</th>
<th>Resource successfully prepared</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_INVALID_PARAMETER</td>
<td>The provided parameter was nullptr</td>
</tr>
</tbody>
</table>
Return values

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR_FILE_NOT_FOUND</td>
<td>The resource file could not be found</td>
</tr>
<tr>
<td>ERROR_COMPRESSION_NOT_AVAILABLE</td>
<td>The selected compression is not available</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_NOT_AVAILABLE</td>
<td>The selected encryption is not available</td>
</tr>
<tr>
<td>ERROR_FILE_OPEN</td>
<td>The file could not be opened</td>
</tr>
<tr>
<td>ERROR_FILE_READ</td>
<td>An error occurred while reading the file</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>p_resCompression</th>
<th>Compression type</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>in</th>
<th>p_resEncryption</th>
<th>Encryption type</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_password</td>
<td>Password for encryption</td>
</tr>
</tbody>
</table>
8.17.3.10  set_error_msg()

```cpp
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 | p_newErrMsg | New error message |

8.17.3.11  set_file()

```cpp
void ResourceData::set_file (const char ∗ p_resFilename )
```

Set file containing resource.

Method to set the filename containing the resource data

Parameters

| in | p_resFilename | Filename containing resource |

8.17.3.12  set_ident()

```cpp
void ResourceData::set_ident (const char ∗ p_resIdent )
```

Set resource identifier.

Method to set the internal resource identifier

Parameters

| in | p_resIdent | Resource identifier |

8.17.3.13  set_rc_position()
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**

<table>
<thead>
<tr>
<th>in</th>
<th>p_line</th>
<th>Line number in RC file</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_col</td>
<td>Column number in RC file</td>
</tr>
</tbody>
</table>

**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 lrc::ResourceManager Class Reference

Manager class handling all resources of a resource file.

#include <ResourceManager.hxx>

Collaboration diagram for lrc::ResourceManager:
Public Member Functions

- **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.
- **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

<table>
<thead>
<tr>
<th>in</th>
<th>p_resFilename</th>
<th>Filename of resource file</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_compress</td>
<td>Compression type if the complete file is compressed</td>
</tr>
<tr>
<td>in</td>
<td>p_encrypt</td>
<td>Encryption type if the complete file is encrypted</td>
</tr>
<tr>
<td>in</td>
<td>p_key</td>
<td>Password if whole file is encrypted</td>
</tr>
</tbody>
</table>

Exceptions

- **lrcFileNotFoundException**  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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_startAddr</code></td>
<td>Start address of resource data in memory</td>
</tr>
<tr>
<td><code>p_endAddr</code></td>
<td>End address of resource data in memory</td>
</tr>
<tr>
<td><code>p_compress</code></td>
<td>Compression type if the complete resource data is compressed</td>
</tr>
<tr>
<td><code>p_encrypt</code></td>
<td>Encryption type if the complete resource data is encrypted</td>
</tr>
<tr>
<td><code>p_key</code></td>
<td>Password if the complete resource is encrypted</td>
</tr>
</tbody>
</table>

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()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_compData</code></td>
<td>Compressed resource data</td>
</tr>
<tr>
<td><code>p_compSize</code></td>
<td>Size of compressed resource data</td>
</tr>
<tr>
<td><code>p_decompData</code></td>
<td>Decompressed resource data</td>
</tr>
<tr>
<td><code>p_decompSize</code></td>
<td>Size of decompressed resource data</td>
</tr>
</tbody>
</table>
Return values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NO_ERROR</strong></td>
<td>Data successfully decompressed</td>
</tr>
<tr>
<td><strong>ERROR_COMPRESSION_DECOMPRESS</strong></td>
<td>An error occurred while decompressing the data</td>
</tr>
</tbody>
</table>

8.18.3.2 decrypt_data()

```cpp
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 method decrypts the resource data in the memory

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>p_encData</strong></td>
<td>Encrypted resource data for decryption</td>
</tr>
<tr>
<td><strong>p_encSize</strong></td>
<td>Size of encrypted resource data</td>
</tr>
<tr>
<td><strong>p_clearData</strong></td>
<td>Decrypted/clear resource data</td>
</tr>
<tr>
<td><strong>p_clearSize</strong></td>
<td>Size of clear resource data</td>
</tr>
</tbody>
</table>

Return values

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NO_ERROR</strong></td>
<td>Data successfully decrypted</td>
</tr>
<tr>
<td><strong>ERROR_ENCRYPTION_NOT_AVAILABLE</strong></td>
<td>Encryption/Decryption support is not compiled in</td>
</tr>
<tr>
<td><strong>ERROR_ENCRYPTION_DECRYPT</strong></td>
<td>An error occurred while decrypting the data</td>
</tr>
</tbody>
</table>

8.18.3.3 get_resource() [1/2]

```cpp
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>p_resID</strong></td>
<td>Resource ID</td>
</tr>
<tr>
<td><strong>p_password</strong></td>
<td>Password if resource is encrypted</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>in</th>
<th>p_resIdx</th>
<th>Index of resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_password</td>
<td>Password if resource is encrypted</td>
</tr>
</tbody>
</table>

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 | p_numRes | Number of resources in list |

Generated by Doxygen
Remarks

The caller is responsible to free the used memory.

8.18.3.6 load_embedded()

```cpp
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

- **p_startAddr**: Start address of resource data in memory
- **p_endAddr**: End address of resource data in memory

Return values

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>Embedded resources successfully accessed</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_NOT_AVAILABLE</td>
<td>Encryption/Decryption support is not compiled in</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_DECRYPT</td>
<td>An error occurred while decrypting the data</td>
</tr>
<tr>
<td>ERROR_COMPRESSION_DECOMPRESS</td>
<td>An error occurred while decompressing the data</td>
</tr>
</tbody>
</table>

8.18.3.7 load_from_file()

```cpp
int lrc::ResourceManager::load_from_file (void ) [private]
```

Method to load resource file.

This method loads all data from the resource file

Return values

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>Resource file successfully loaded</td>
</tr>
<tr>
<td>ERROR_FILE_READ</td>
<td>An error occurred while reading the file</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_NOT_AVAILABLE</td>
<td>Encryption/Decryption support is not compiled in</td>
</tr>
<tr>
<td>ERROR_ENCRYPTION_DECRYPT</td>
<td>An error occurred while decrypting the data</td>
</tr>
<tr>
<td>ERROR_COMPRESSION_DECOMPRESS</td>
<td>An error occurred while decompressing the data</td>
</tr>
</tbody>
</table>
8.18.3.8  setup_resources()

```cpp
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

<table>
<thead>
<tr>
<th>in</th>
<th>p_resData</th>
<th>Decrypted and decompressed resource data</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_resSize</td>
<td>Size of resource data</td>
</tr>
</tbody>
</table>

Return values

- **NO_ERROR** 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

```cpp
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 (lrcFileNotFoundException)**
  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()

```cpp
RIFParser::RIFParser ( 
  char ∗ p_filename ) throw lrcFileNotFoundException)
```

Constructor.

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

**Parameters**

| in | p_filename | Filename of .rif file |

**Exceptions**

| lrcFileNotFoundException | Exception that is thrown if the given .rif file could not be found |

8.19.3 Member Function Documentation

8.19.3.1 parse()

```cpp
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ERROR</td>
<td>File successfully parsed</td>
</tr>
<tr>
<td>ERROR_FILE_OPEN</td>
<td>An error occurred while trying to open the file</td>
</tr>
<tr>
<td>ERRORPARSE</td>
<td>An error occurred while trying to parse the file</td>
</tr>
</tbody>
</table>

Implements InFileParser.

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

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

8.20 SerpentEncryption Class Reference

Class to encrypt/decrypt using the Serpent algorithm.

#include <SerpentEncryption.hxx>

Inheritance diagram for SerpentEncryption:
Collaboration diagram for SerpentEncryption:

![Collaboration Diagram]

### 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()

```cpp
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>p_ivArray</code></td>
<td>Array for initialization vector</td>
</tr>
<tr>
<td><code>p_ivSize</code></td>
<td>Size of the array</td>
</tr>
</tbody>
</table>

### Return values

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NO_ERROR</code></td>
<td>Initialization vector successfully created</td>
</tr>
<tr>
<td><code>ERROR_INVALID_PARAMETER</code></td>
<td>The array was not created</td>
</tr>
</tbody>
</table>

#### 8.20.2.2 decrypt()

```cpp
defrypt (               
    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 [lrc::EncryptDecrypt](#).

#### 8.20.2.3 encrypt()

```cpp
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 [lrc::EncryptDecrypt](#).

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

- `/home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx`
8.21 zLibCompression Class Reference

Compression class that uses the zLib algorithm.

#include <zLibCompression.hxx>

Inheritance diagram for zLibCompression:

```
   Irc::CompressDecompress
       
          zLibCompression
```

Collaboration diagram for zLibCompression:

```
   Irc::CompressDecompress
       
          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()

```cpp
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 `lrc::CompressDecompress`.

8.21.2.2 decompress()

```cpp
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 `lrc::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"
9.2 /home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx File Reference

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

```cpp
#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 Tscharner

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()

```c
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

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

Include dependency graph for ParserFactory.hxx:

```
Generated by Doxygen
```
This graph shows which files directly or indirectly include this file:

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

Classes

- class ParserFactory

  Factory class to create an appropriate parser class.

9.4.1 Detailed Description

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

Author

Andreas Tscharner

Date

2014-08-23

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

```c
#include <regex>
#include "InFileParser.hxx"
```
Include dependency graph for RCParser.hxx:

Classes

- class RCParser

  Class to parse an .rc file.

9.5.1 Detailed Description

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

Author

Andreas Tscharner

Date

2014-10-26
Classes

- class RIFParser
  
  Class to parse a .rif (XML) file.

9.6.1 Detailed Description

This file contains the declaration of the RIFParser class, a class to parse the RIF input file for lrc

Author

Andreas Tscharner

Date

2014-08-23
#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 Tscharner

Date

2014-08-23
#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 Tscharner

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:

```
/home/andy/Programming/Projects/lrc/src/include/Resource.hxx
/home/andy/Programming/Projects/lrc/src/ResourceData.hxx
/home/andy/Programming/Projects/lrc/src/include/ResourceManager.hxx
/home/andy/Programming/Projects/lrc/src/compiler/Collector.hxx
/home/andy/Programming/Projects/lrc/src/compiler/InFileParser.hxx
/home/andy/Programming/Projects/lrc/src/compiler/lrc.cxx
/home/andy/Programming/Projects/lrc/src/compiler/ParserFactory.hxx
/home/andy/Programming/Projects/lrc/src/compiler/RCParser.hxx
/home/andy/Programming/Projects/lrc/src/compiler/RIFParser.hxx
```

**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_.

Generated by Doxygen
9.10.1 Detailed Description

This file contains one of the main classes: Resource

Author

Andreas Tscharner

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 Tscharner

Date

2014-08-23
9.12 /home/andy/Programming/Projects/lrc/src/lrcExceptions.hxx File Reference

```cpp
#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
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 Tscharner

Date

2014-08-02

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
### 9.14.3.2 is_warning()

```cpp
bool is_warning ( const int c ) [inline]
```

Check for warning.

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

Parameters

<table>
<thead>
<tr>
<th>in c</th>
<th>Value to check</th>
</tr>
</thead>
</table>

Return values

| true | Value is a warning code |
| false | Value stands for success or error |

Definition at line 104 of file StatusCodes.hxx.

### 9.14.3.3 no_error()

```cpp
bool no_error ( const int c ) [inline]
```

Check for NO_ERROR.

Small inline function to check if the given parameter is NO_ERROR

Parameters

<table>
<thead>
<tr>
<th>in c</th>
<th>Value to check</th>
</tr>
</thead>
</table>
Return values

<table>
<thead>
<tr>
<th>true</th>
<th>Value is NO_ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>Value is not NO_ERROR</td>
</tr>
</tbody>
</table>

Definition at line 81 of file StatusCodes.hxx.

9.14.3.4 success()

```c
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

| in  | c  | Value to check |

Return values

<table>
<thead>
<tr>
<th>true</th>
<th>Value means success</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>Value means warning or error</td>
</tr>
</tbody>
</table>

Definition at line 92 of file StatusCodes.hxx.

9.15

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

File Reference

#include "../include/CompressDecompress.hxx"
Include dependency graph for bz2LibCompression.hxx:

```
/home/andy/Programming/Projects/lrc/src/strategies/bz2LibCompression.hxx
../include/CompressDecompress.hxx
stddef.h
```

### Classes

- class **bz2LibCompression**

  Compression class that uses the bzip2 algorithm.

#### 9.15.1 Detailed Description

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

**Author**

Andreas Tscharner

**Date**

2012-01-18

### 9.16 Include Reference

```
#include "../include/CompressDecompress.hxx"
```
Include dependency graph for NoneCompression.hxx:

```
/home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.hxx
../include/CompressDecompress.hxx
stddef.h
```

Classes

- class **NoneCompression**
  
  *Compression class that does NO compression.*

9.16.1 Detailed Description

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

Author

Andreas Tscharner

Date

2011-06-25

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

`#include "../include/EncryptDecrypt.hxx"`
Include dependency graph for NoneEncryption.hxx:

![Include dependency graph for NoneEncryption.hxx](image)

### Classes

- class **NoneEncryption**

  *Encryption class that does NO encryption.*

### 9.17.1 Detailed Description

This file contains the **NoneEncryption** class. This class does no encryption at all, but is used to fit in the Strategy Pattern

**Author**

Andreas Tscharner

**Date**

0211-07-21

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

```
#include "../include/EncryptDecrypt.hxx"
```
Include dependency graph for SerpentEncryption.hxx:

```
/home/andy/Programming/Projects/lrc/src/strategies/SerpentEncryption.hxx
../include/EncryptDecrypt.hxx
stddef.h

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 Tscharner

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:

```
/home/andy/Programming/Projects/lrc/src/strategies/zLibCompression.hxx

../include/CompressDecompress.hxx

stddef.h
```

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

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

```
/home/andy/Programming
Projects/lrc/src/Utils.hxx
/home/andy/Programming
Projects/lrc/src/compiler
/lrc.cxx
```

**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 Tscharner

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

<table>
<thead>
<tr>
<th>in</th>
<th>p_dataList</th>
<th>List of data to delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_dataSize</td>
<td>Size of data list</td>
</tr>
</tbody>
</table>
9.20.3.2 file_exists()

```cpp
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**

- `p_filename`: Full or relative path to file

**Return values**

- `true`: File exists
- `false`: File does not exist

9.20.3.3 file_size()

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

Returns the size of a file.

This function returns the size of a file.

**Parameters**

- `p_filename`: Filename

**Returns**

- Size of file

**Remarks**

If an error occurs, the function returns -1

9.20.3.4 get_extension()

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

Returns the file extension.

This function returns the file extension of the given file.
9.20 File Reference

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_filename</td>
</tr>
</tbody>
</table>

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()

```c
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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_password</td>
</tr>
</tbody>
</table>

Returns

Length of zero-terminated password

Remarks

-1 will be returned if the given password was nullptr

9.20.3.6 replace_extension()

```c
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 extention is expected to have a dot (".") as first character. If the given filename has no extension, the new extension is appended.
Parameters

<table>
<thead>
<tr>
<th>in</th>
<th>p_filename</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>in</td>
<td>p_newExt</td>
<td>New file extension</td>
</tr>
</tbody>
</table>

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/RCParser.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/bz2LibCompression.hxx, 107
/home/andy/Programming/Projects/lrc/src/strategies/NoneCompression.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

~IrcEncryptionDisabledException
  IrcEncryptionDisabledException, 36
~IrcFileExistsException
  IrcFileExistsException, 39
~IrcFileNotFoundException
  IrcFileNotFoundException, 41

are_resIDs_unique
  Collector, 18

bz2LibCompression, 15
  compress, 16
decompress, 16

clear_internal_error
  InFileParser, 31
collect
  Collector, 19
  ~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
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
lrc.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
lrc.cxx, 85

WARNING_BASE
StatusCodes.hxx, 104
WARNING_DOUBLE_RESOURCE_ID
StatusCodes.hxx, 104

what
lrcEncryptionDisabledException, 37
lrcFileExistsException, 39
lrcFileNotFoundException, 42

zLibCompression, 78
compress, 79
decompress, 79