ISA LDAP server

Generated by Doxygen 1.9.8

1 Hierarchical Index 1

1.1 Class Hierarchy . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 AndFilter Class Reference . . . 7

4.1.1 Detailed Description . . . 8

4.1.2 Constructor & Destructor Documentation . . . 8

4.1.2.1 ~AndFilter() . . . 8

4.1.3 Member Function Documentation . . . 8

4.1.3.1 getFilterType() . . . 8

4.1.4 Member Data Documentation . . . 8

4.1.4.1 filters . . . 8

4.2 args_t Struct Reference . . . 8

4.2.1 Detailed Description . . . 9

4.2.2 Member Data Documentation . . . 9

4.2.2.1 dbPath . . . 9

4.2.2.2 err . . . 9

4.2.2.3 port . . . 9

4.3 BerBoolObject Class Reference . . . 9

4.3.1 Detailed Description . . . 10

4.3.2 Constructor & Destructor Documentation . . . 10

4.3.2.1 BerBoolObject() . . . 10

4.3.2.2 ~BerBoolObject() . . . 10

4.3.3 Member Function Documentation . . . 11

4.3.3.1 getBerObjectType() . . . 11

4.3.3.2 getBerRepresentation() . . . 11

4.3.3.3 getLenght() . . . 11

4.4 BerEnumObject Class Reference . . . 12

4.4.1 Detailed Description . . . 12

4.4.2 Constructor & Destructor Documentation . . . 13

4.4.2.1 BerEnumObject() . . . 13

4.4.2.2 ~BerEnumObject() . . . 13

4.4.3 Member Function Documentation . . . 13

4.4.3.1 getBerObjectType() . . . 13

4.4.3.2 getBerRepresentation() . . . 13

4.4.3.3 getLenght() . . . 14

4.5 BerIntObject Class Reference . . . 14

4.5.1 Detailed Description . . . 15
4.5.2 Constructor & Destructor Documentation . . . 15
4.5.2.1 BerIntObject() [1/2] . . . 15
4.5.2.2 BerIntObject() [2/2] . . . 15
4.5.2.3 ~BerIntObject() . . . 15
4.5.3 Member Function Documentation . . . 15
4.5.3.1 getBerObjectType() . . . 15
4.5.3.2 getBerRepresentation() . . . 16
4.5.3.3 getLenght() . . . 16
4.5.3.4 getValue() . . . 17
4.5.3.5 setValue() . . . 17
4.6 BerObject Class Reference . . . 17
4.6.1 Detailed Description . . . 18
4.6.2 Constructor & Destructor Documentation . . . 18
4.6.2.1 ~BerObject() . . . 18
4.6.3 Member Function Documentation . . . 18
4.6.3.1 getBerObjectType() . . . 18
4.6.3.2 getBerRepresentation() . . . 19
4.6.3.3 getLenght() . . . 19
4.7 BerSequenceObject Class Reference . . . 20
4.7.1 Detailed Description . . . 21
4.7.2 Constructor & Destructor Documentation . . . 21
4.7.2.1 BerSequenceObject() [1/2] . . . 21
4.7.2.2 BerSequenceObject() [2/2] . . . 21
4.7.2.3 ~BerSequenceObject() . . . 21
4.7.3 Member Function Documentation . . . 21
4.7.3.1 getBerObjectType() . . . 21
4.7.3.2 getBerRepresentation() . . . 22
4.7.3.3 getLenght() . . . 22
4.7.3.4 GetTag() . . . 23
4.7.4 Member Data Documentation . . . 23
4.7.4.1 objects . . . 23
4.8 BerSetObject Class Reference . . . 23
4.8.1 Detailed Description . . . 24
4.8.2 Constructor & Destructor Documentation . . . 24
4.8.2.1 BerSetObject() . . . 24
4.8.2.2 ~BerSetObject() . . . 24
4.8.3 Member Function Documentation . . . 24
4.8.3.1 getBerObjectType() . . . 24
4.8.3.2 getBerRepresentation() . . . 25
4.8.3.3 getLenght() . . . 25
4.8.4 Member Data Documentation . . . 25

4.8.4.1 objects . . . 25
4.9 BerStringObject Class Reference . . . 26
4.9.1 Detailed Description . . . 27
4.9.2 Constructor & Destructor Documentation . . . 27
4.9.2.1 BerStringObject() [1/3] . . . 27
4.9.2.2 BerStringObject() [2/3] . . . 27
4.9.2.3 BerStringObject() [3/3] . . . 27
4.9.3 Member Function Documentation . . . 27
4.9.3.1 getBerObjectType() . . . 27
4.9.3.2 getBerRepresentation() . . . 28
4.9.3.3 getLenght() . . . 28
4.9.4 Member Data Documentation . . . 28
4.9.4.1 value . . . 28
4.10 BerUndefinedObject Class Reference . . . 29
4.10.1 Detailed Description . . . 29
4.10.2 Constructor & Destructor Documentation . . . 30
4.10.2.1 BerUndefinedObject() . . . 30
4.10.3 Member Function Documentation . . . 30
4.10.3.1 getBerObjectType() . . . 30
4.10.3.2 getBerRepresentation() . . . 30
4.10.3.3 getLenght() . . . 31
4.11 DatabaseController Class Reference . . . 31
4.11.1 Detailed Description . . . 32
4.11.2 Constructor & Destructor Documentation . . . 32
4.11.2.1 DatabaseController() . . . 32
4.11.2.2 ∼DatabaseController() . . . 32
4.11.3 Member Function Documentation . . . 32
4.11.3.1 loadAllRows() . . . 32
4.11.3.2 loadNextRow() . . . 33
4.12 DatabaseObject Class Reference . . . 33
4.12.1 Detailed Description . . . 34
4.12.2 Constructor & Destructor Documentation . . . 34
4.12.2.1 DatabaseObject() . . . 34
4.12.3 Member Function Documentation . . . 34
4.12.3.1 get_email() . . . 34
4.12.3.2 get_name() . . . 34
4.12.3.3 get_uid() . . . 34
4.13 EqualityMatchFilter Class Reference . . . 35
4.13.1 Detailed Description . . . 35
4.13.2 Constructor & Destructor Documentation . . . 36
4.13.2.1 EqualityMatchFilter() . . . 36
4.13.3 Member Function Documentation . . . 36

4.13.3.1 getAssertionValue() . . . 36
4.13.3.2 getAttributeDescription() . . . 36
4.13.3.3 getFilterType() . . . 36
4.14 FilterObject Class Reference . . . 36
4.14.1 Detailed Description . . . 37
4.14.2 Constructor & Destructor Documentation . . . 37
4.14.2.1 ~FilterObject() . . . 37
4.14.3 Member Function Documentation . . . 37
4.14.3.1 getFilterType() . . . 37
4.15 NotFilter Class Reference . . . 37
4.15.1 Detailed Description . . . 38
4.15.2 Constructor & Destructor Documentation . . . 38
4.15.2.1 ~NotFilter() . . . 38
4.15.3 Member Function Documentation . . . 38
4.15.3.1 getFilterType() . . . 38
4.15.4 Member Data Documentation . . . 39
4.15.4.1 filter . . . 39
4.16 OrFilter Class Reference . . . 39
4.16.1 Detailed Description . . . 40
4.16.2 Constructor & Destructor Documentation . . . 40
4.16.2.1 ~OrFilter() . . . 40
4.16.3 Member Function Documentation . . . 40
4.16.3.1 getFilterType() . . . 40
4.16.4 Member Data Documentation . . . 40
4.16.4.1 filters . . . 40
4.17 searchedAttributes Struct Reference . . . 40
4.17.1 Detailed Description . . . 41
4.17.2 Member Data Documentation . . . 41
4.17.2.1 cn . . . 41
4.17.2.2 email . . . 41
4.17.2.3 uid . . . 41
4.18 searchRequest Struct Reference . . . 41
4.18.1 Detailed Description . . . 42
4.18.2 Member Data Documentation . . . 42
4.18.2.1 attributes . . . 42
4.18.2.2 messageIDLength . . . 42
4.18.2.3 sizeLimit . . . 42
4.19 SubstringFilter Class Reference . . . 43
4.19.1 Detailed Description . . . 43
4.19.2 Constructor & Destructor Documentation . . . 44
4.19.2.1 SubstringFilter() . . . 44
4.19.3 Member Function Documentation . . . 44

4.19.3.1 getAttributeDescription() . . . 44
4.19.3.2 getFilterType() . . . 44
4.19.3.3 getSubAny() . . . 44
4.19.3.4 getSubFinal() . . . 44
4.19.3.5 getSubInitial() . . . 44

5 File Documentation **45**

5.1 inc/AndFilterObject.h File Reference . . . 45
5.1.1 Detailed Description . . . 46
5.2 AndFilterObject.h . . . 46
5.3 inc/argument_helper_functions.h File Reference . . . 47
5.3.1 Detailed Description . . . 48
5.3.2 Function Documentation . . . 48
5.3.2.1 parseArguments() . . . 48
5.4 argument_helper_functions.h . . . 48
5.5 inc/ber_constants.h File Reference . . . 49
5.5.1 Detailed Description . . . 50
5.5.2 Variable Documentation . . . 50
5.5.2.1 BER_4BYTE_LENGTH_LENGTH . . . 50
5.5.2.2 BER_BIND_REQUEST_C . . . 50
5.5.2.3 BER_BIND_RESPONSE_C . . . 50
5.5.2.4 BER_BOOL_C . . . 50
5.5.2.5 BER_ENUM_C . . . 50
5.5.2.6 BER_EXTENDED_RESPONSE_C . . . 51
5.5.2.7 BER_INT_4BYTES_C . . . 51
5.5.2.8 BER_INT_C . . . 51
5.5.2.9 BER_LDAP_AUTH_METHOD_NOT_SUPPORTED . . . 51
5.5.2.10 BER_LDAP_PROTOCOL_ERROR . . . 51
5.5.2.11 BER_LDAP_SIZE_LIMIT_EXCEEDED . . . 51
5.5.2.12 BER_LDAP_SUCCESS . . . 51
5.5.2.13 BER_LDAP_UNAVAILABLE . . . 51
5.5.2.14 BER_LENGTH_OF_LENGTH_TAG . . . 52
5.5.2.15 BER_OCTET_STRING_C . . . 52
5.5.2.16 BER_SEARCH_REQUEST_C . . . 52
5.5.2.17 BER_SEARCH_RESULT_DONE_C . . . 52
5.5.2.18 BER_SEARCH_RESULT_ENTRY_C . . . 52
5.5.2.19 BER_SEQUENCE_C . . . 52
5.5.2.20 BER_SET_C . . . 52
5.5.2.21 BER_TAG_LENGTH . . . 52
5.5.2.22 BER_UNBIND_REQUEST_C . . . 53
5.6 ber_constants.h . . . 53
5.7 inc/ber_helper_functions.h File Reference . . . 53

5.7.1 Detailed Description . . . 55
5.7.2 Typedef Documentation . . . 55
5.7.2.1 filterTypes . . . 55
5.7.3 Enumeration Type Documentation . . . 55
5.7.3.1 berObjectTypes . . . 55
5.7.3.2 filterTypes . . . 55
5.7.4 Function Documentation . . . 56
5.7.4.1 AppendLenght4Bytes() . . . 56
5.7.4.2 getFilterType() . . . 56
5.7.4.3 GetLength() . . . 57
5.7.4.4 GetLengthOfLength() . . . 58
5.7.4.5 GoIntoTag() . . . 59
5.7.4.6 HowManyBytesWillIntUse() . . . 60
5.7.4.7 IncreaseLength4Bytes() . . . 60
5.7.4.8 ParseINT() [1/2] . . . 61
5.7.4.9 ParseINT() [2/2] . . . 62
5.7.4.10 SkipTags() . . . 63
5.7.4.11 ToLowerCase() . . . 63
5.7.4.12 WriteIntAppend() . . . 64
5.8 ber_helper_functions.h . . . 65
5.9 inc/BerBoolObject.h File Reference . . . 65
5.9.1 Detailed Description . . . 66
5.10 BerBoolObject.h . . . 67
5.11 inc/BerEnumObject.h File Reference . . . 67
5.11.1 Detailed Description . . . 68
5.12 BerEnumObject.h . . . 68
5.13 inc/BerIntObject.h File Reference . . . 69
5.13.1 Detailed Description . . . 70
5.14 BerIntObject.h . . . 70
5.15 inc/BerObject.h File Reference . . . 70
5.15.1 Detailed Description . . . 71
5.16 BerObject.h . . . 71
5.17 inc/BerParser.h File Reference . . . 72
5.17.1 Detailed Description . . . 73
5.17.2 Function Documentation . . . 73
5.17.2.1 ParseBerObject() . . . 73
5.18 BerParser.h . . . 74
5.19 inc/BerSequenceObject.h File Reference . . . 75
5.19.1 Detailed Description . . . 76
5.20 BerSequenceObject.h . . . 76
5.21 inc/BerSetObject.h File Reference . . . 76
5.21.1 Detailed Description . . . 77

5.22 BerSetObject.h . . . 78
5.23 inc/BerStringObject.h File Reference . . . 78
5.23.1 Detailed Description . . . 79
5.24 BerStringObject.h . . . 79
5.25 inc/BerUndefinedObject.h File Reference . . . 80
5.25.1 Detailed Description . . . 81
5.26 BerUndefinedObject.h . . . 81
5.27 inc/database_helper_functions.h File Reference . . . 81
5.27.1 Detailed Description . . . 82
5.27.2 Function Documentation . . . 83
5.27.2.1 removeDuplicates() . . . 83
5.28 database_helper_functions.h . . . 83
5.29 inc/DatabaseController.h File Reference . . . 84
5.29.1 Detailed Description . . . 85
5.30 DatabaseController.h . . . 85
5.31 inc/DatabaseObject.h File Reference . . . 85
5.31.1 Detailed Description . . . 86
5.32 DatabaseObject.h . . . 87
5.33 inc/EqualityMatchFilterObject.h File Reference . . . 87
5.33.1 Detailed Description . . . 88
5.34 EqualityMatchFilterObject.h . . . 88
5.35 inc/filter_helper_functions.h File Reference . . . 89
5.35.1 Detailed Description . . . 90
5.35.2 Function Documentation . . . 90
5.35.2.1 convertToFilterObject() . . . 90
5.35.2.2 equalityMatchHandler() . . . 91
5.35.2.3 filterHandler() . . . 92
5.35.2.4 filterLine() . . . 93
5.35.2.5 substrFilterHandler() . . . 94
5.36 filter_helper_functions.h . . . 95
5.37 inc/FilterObject.h File Reference . . . 95
5.37.1 Detailed Description . . . 96
5.38 FilterObject.h . . . 97
5.39 inc/ldap_comunication.h File Reference . . . 97
5.39.1 Detailed Description . . . 99
5.39.2 Enumeration Type Documentation . . . 99
5.39.2.1 atributeDescriptions . . . 99
5.39.3 Function Documentation . . . 99
5.39.3.1 AddToSearchResultEntry() . . . 99
5.39.3.2 checkSearchRequest() . . . 100
5.39.3.3 CreateBindResponse() . . . 100
5.39.3.4 InitSearchResultEntry() . . . 101

5.39.3.5 loadEnvelope() . . . 102
5.39.3.6 searchRequestHandler() . . . 102
5.39.3.7 sendNoticeOfDisconnection() . . . 103
5.39.3.8 sendSearchResultDone() . . . 104
5.40 ldap_comunication.h . . . 105
5.41 inc/NotFilterObject.h File Reference . . . 106
5.41.1 Detailed Description . . . 107
5.42 NotFilterObject.h . . . 108
5.43 inc/OrFilterObject.h File Reference . . . 108
5.43.1 Detailed Description . . . 109
5.44 OrFilterObject.h . . . 109
5.45 inc/server.h File Reference . . . 110
5.45.1 Detailed Description . . . 111
5.45.2 Macro Definition Documentation . . . 111
5.45.2.1 CHECK_ERR . . . 111
5.45.3 Function Documentation . . . 111
5.45.3.1 ldapServer() . . . 111
5.46 server.h . . . 112
5.47 inc/SubstringFilterObject.h File Reference . . . 113
5.47.1 Detailed Description . . . 114
5.48 SubstringFilterObject.h . . . 114
5.49 src/AndFilterObject.cpp File Reference . . . 115
5.49.1 Detailed Description . . . 115
5.50 AndFilterObject.cpp . . . 116
5.51 src/argument_helper_functions.cpp File Reference . . . 116
5.51.1 Detailed Description . . . 116
5.51.2 Function Documentation . . . 117
5.51.2.1 parseArguments() . . . 117
5.52 argument_helper_functions.cpp . . . 117
5.53 src/ber_helper_functions.cpp File Reference . . . 118
5.53.1 Detailed Description . . . 119
5.53.2 Function Documentation . . . 119
5.53.2.1 AppendLenght4Bytes() . . . 119
5.53.2.2 getFilterType() . . . 119
5.53.2.3 GetLength() . . . 120
5.53.2.4 GetLengthOfLength() . . . 121
5.53.2.5 GoIntoTag() . . . 122
5.53.2.6 HowManyBytesWillIntUse() . . . 123
5.53.2.7 IncreaseLength4Bytes() . . . 123
5.53.2.8 ParseINT() . . . 124
5.53.2.9 SkipTags() . . . 125
5.53.2.10 ToLowerCase() . . . 126

5.53.2.11 WriteIntAppend() . . . 127
5.54 ber_helper_functions.cpp . . . 127
5.55 src/BerBoolObject.cpp File Reference . . . 131
5.55.1 Detailed Description . . . 131
5.56 BerBoolObject.cpp . . . 131
5.57 src/BerEnumObject.cpp File Reference . . . 132
5.57.1 Detailed Description . . . 132
5.58 BerEnumObject.cpp . . . 133
5.59 src/BerIntObject.cpp File Reference . . . 133
5.59.1 Detailed Description . . . 133
5.60 BerIntObject.cpp . . . 134
5.61 src/BerObject.cpp File Reference . . . 134
5.61.1 Detailed Description . . . 135
5.62 BerObject.cpp . . . 135
5.63 src/BerParser.cpp File Reference . . . 135
5.63.1 Detailed Description . . . 136
5.63.2 Function Documentation . . . 136
5.63.2.1 ParseBerObject() . . . 136
5.64 BerParser.cpp . . . 137
5.65 src/BerSequenceObject.cpp File Reference . . . 139
5.65.1 Detailed Description . . . 139
5.66 BerSequenceObject.cpp . . . 140
5.67 src/BerSetObject.cpp File Reference . . . 141
5.67.1 Detailed Description . . . 141
5.68 BerSetObject.cpp . . . 141
5.69 src/BerStringObject.cpp File Reference . . . 142
5.69.1 Detailed Description . . . 142
5.70 BerStringObject.cpp . . . 143
5.71 src/BerUndefinedObject.cpp File Reference . . . 143
5.71.1 Detailed Description . . . 144
5.72 BerUndefinedObject.cpp . . . 144
5.73 src/database_helper_functions.cpp File Reference . . . 144
5.73.1 Detailed Description . . . 145
5.73.2 Function Documentation . . . 145
5.73.2.1 removeDuplicates() . . . 145
5.74 database_helper_functions.cpp . . . 146
5.75 src/DatabaseController.cpp File Reference . . . 146
5.75.1 Detailed Description . . . 146
5.76 DatabaseController.cpp . . . 147
5.77 src/DatabaseObject.cpp File Reference . . . 148
5.77.1 Detailed Description . . . 148
5.78 DatabaseObject.cpp . . . 148

5.79 src/EqualityMatchFilterObject.cpp File Reference . . . 149
5.79.1 Detailed Description . . . 149
5.80 EqualityMatchFilterObject.cpp . . . 149
5.81 src/filter_helper_functions.cpp File Reference . . . 150
5.81.1 Detailed Description . . . 150
5.81.2 Function Documentation . . . 150
5.81.2.1 convertToFilterObject() . . . 150
5.81.2.2 equalityMatchHandler() . . . 151
5.81.2.3 filterHandler() . . . 152
5.81.2.4 filterLine() . . . 153
5.81.2.5 substrFilterHandler() . . . 154
5.82 filter_helper_functions.cpp . . . 155
5.83 src/FilterObject.cpp File Reference . . . 159
5.83.1 Detailed Description . . . 160
5.84 FilterObject.cpp . . . 160
5.85 src/isa-ldapserver.cpp File Reference . . . 160
5.85.1 Detailed Description . . . 160
5.85.2 Function Documentation . . . 161
5.85.2.1 file_exists() . . . 161
5.85.2.2 main() . . . 161
5.86 isa-ldapserver.cpp . . . 161
5.87 src/ldap_comunication.cpp File Reference . . . 161
5.87.1 Detailed Description . . . 162
5.87.2 Macro Definition Documentation . . . 162
5.87.2.1 DEBUG . . . 162
5.87.3 Function Documentation . . . 162
5.87.3.1 AddToSearchResultEntry() . . . 162
5.87.3.2 checkSearchRequest() . . . 163
5.87.3.3 CreateBindResponse() . . . 164
5.87.3.4 InitSearchResultEntry() . . . 164
5.87.3.5 loadEnvelope() . . . 165
5.87.3.6 searchRequestHandler() . . . 166
5.87.3.7 sendNoticeOfDisconnection() . . . 167
5.87.3.8 sendSearchResultDone() . . . 167
5.88 ldap_comunication.cpp . . . 168
5.89 src/NotFilterObject.cpp File Reference . . . 172
5.89.1 Detailed Description . . . 173
5.90 NotFilterObject.cpp . . . 173
5.91 src/OrFilterObject.cpp File Reference . . . 173
5.91.1 Detailed Description . . . 174
5.92 OrFilterObject.cpp . . . 174
5.93 src/server.cpp File Reference . . . 174

5.93.1 Detailed Description . . . 175
5.93.2 Macro Definition Documentation . . . 175
5.93.2.1 CLEANUP_SERVER . . . 175
5.93.3 Function Documentation . . . 175
5.93.3.1 ldapServer() . . . 175
5.93.3.2 SigCatcher() . . . 176
5.93.3.3 SigIntCatcher() . . . 176
5.93.3.4 SigQuitCatcher() . . . 177
5.93.4 Variable Documentation . . . 177
5.93.4.1 children . . . 177
5.93.4.2 childSocket . . . 178
5.93.4.3 communicationSocket . . . 178
5.94 server.cpp . . . 178
5.95 src/SubstringFilterObject.cpp File Reference . . . 181
5.95.1 Detailed Description . . . 181
5.96 SubstringFilterObject.cpp . . . 181

Index **183**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

- args_t . . . 8
- BerObject . . . 17
  - BerBoolObject . . . 9
  - BerEnumObject . . . 12
  - BerIntObject . . . 14
  - BerSequenceObject . . . 20
  - BerSetObject . . . 23
  - BerStringObject . . . 26
  - BerUndefinedObject . . . 29
- DatabaseController . . . 31
- DatabaseObject . . . 33
- FilterObject . . . 36
  - AndFilter . . . 7
  - EqualityMatchFilter . . . 35
  - NotFilter . . . 37
  - OrFilter . . . 39
  - SubstringFilter . . . 43
- searchedAttributes . . . 40
- searchRequest . . . 41

Chapter 2

Class Index

2.1 Class List

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

AndFilter . . . 7
args_t . . . 8
BerBoolObject . . . 9
BerEnumObject . . . 12
BerIntObject . . . 14
BerObject
    Base class for all BER objects . . . 17
BerSequenceObject . . . 20
BerSetObject . . . 23
BerStringObject . . . 26
BerUndefinedObject . . . 29
DatabaseController
    Class for loading and parsing database file . . . 31
DatabaseObject
    Object representing one row from database . . . 33
EqualityMatchFilter . . . 35
FilterObject
    Base class for all filter objects . . . 36
NotFilter . . . 37
OrFilter . . . 39
searchedAttributes . . . 40
searchRequest . . . 41
SubstringFilter . . . 43

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

inc/AndFilterObject.h
: And filter object for BER LDAP . . . 45

inc/argument_helper_functions.h
: Helper functions for parsing arguments . . . 47

inc/ber_constants.h
: Constants that are used in LDAP BER . . . 49

inc/ber_helper_functions.h
: Helper functions for parsing BER LDAP . . . 53

inc/BerBoolObject.h
: Boolean object for BER LDAP . . . 65

inc/BerEnumObject.h
: Enum object for BER LDAP . . . 67

inc/BerIntObject.h
: Integer object for BER LDAP . . . 69

inc/BerObject.h
: Base class for all BER objects . . . 70

inc/BerParser.h
: Parser for BER LDAP . . . 72

inc/BerSequenceObject.h
: Sequence object for BER LDAP . . . 75

inc/BerSetObject.h
: Set object for BER LDAP . . . 76

inc/BerStringObject.h
: String object for BER LDAP . . . 78

inc/BerUndefinedObject.h
: Undefined object for BER LDAP, for containing unknown data . . . 80

inc/database_helper_functions.h
: Helper functions for database . . . 81

inc/DatabaseController.h
: Controller for database csv file . . . 84

inc/DatabaseObject.h
: Object representing one row from database . . . 85

inc/EqualityMatchFilterObject.h
: Equality match filter object for BER LDAP . . . 87

inc/filter_helper_functions.h
: Helper functions for filters . . . 89

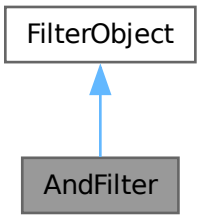
inc/FilterObject.h
Base class for all filter objects . . . 95
inc/ldap_comunication.h
Functions for communication with ldap client . . . 97
inc/NotFilterObject.h
Helper functions for parsing arguments . . . 106
inc/OrFilterObject.h
Object for OR filter . . . 108
inc/server.h
Ldap server implementation . . . 110
inc/SubstringFilterObject.h
Object for substring filter . . . 113
src/AndFilterObject.cpp . . . 115
src/argument_helper_functions.cpp . . . 116
src/ber_helper_functions.cpp . . . 118
src/BerBoolObject.cpp . . . 131
src/BerEnumObject.cpp . . . 132
src/BerIntObject.cpp . . . 133
src/BerObject.cpp . . . 134
src/BerParser.cpp . . . 135
src/BerSequenceObject.cpp . . . 139
src/BerSetObject.cpp . . . 141
src/BerStringObject.cpp . . . 142
src/BerUndefinedObject.cpp . . . 143
src/database_helper_functions.cpp . . . 144
src/DatabaseController.cpp . . . 146
src/DatabaseObject.cpp . . . 148
src/EqualityMatchFilterObject.cpp . . . 149
src/filter_helper_functions.cpp . . . 150
src/FilterObject.cpp . . . 159
src/isa-ldapserver.cpp . . . 160
src/ldap_comunication.cpp . . . 161
src/NotFilterObject.cpp . . . 172
src/OrFilterObject.cpp . . . 173
src/server.cpp . . . 174
src/SubstringFilterObject.cpp . . . 181

Chapter 4

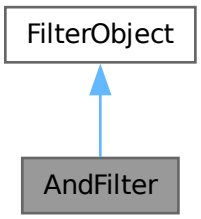
Class Documentation

4.1 AndFilter Class Reference

Inheritance diagram for AndFilter:



Collaboration diagram for AndFilter:



Public Member Functions

- filterTypes getFilterType ()

Public Attributes

- std::vector< FilterObject * > filters

4.1.1 Detailed Description

Definition at line 14 of file AndFilterObject.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ~AndFilter()

```
AndFilter::~AndFilter ( )
```

Definition at line 10 of file AndFilterObject.cpp.

4.1.3 Member Function Documentation

4.1.3.1 getFilterType()

```
filterTypes AndFilter::getFilterType ( )  [virtual]
```

Reimplemented from FilterObject.

Definition at line 9 of file AndFilterObject.cpp.

4.1.4 Member Data Documentation

4.1.4.1 filters

```
std::vector<FilterObject *> AndFilter::filters
```

Definition at line 16 of file AndFilterObject.h.

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

- inc/AndFilterObject.h
- src/AndFilterObject.cpp

4.2 args_t Struct Reference

Public Attributes

- char * dbPath
- int port
- bool err

4.2.1 Detailed Description

Definition at line 13 of file argument_helper_functions.h.

4.2.2 Member Data Documentation

4.2.2.1 dbPath

```
char* args_t::dbPath
```

Definition at line 14 of file argument_helper_functions.h.

4.2.2.2 err

```
bool args_t::err
```

Definition at line 16 of file argument_helper_functions.h.

4.2.2.3 port

```
int args_t::port
```

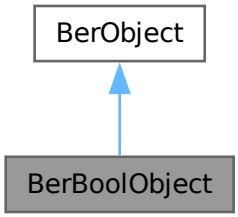
Definition at line 15 of file argument_helper_functions.h.

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

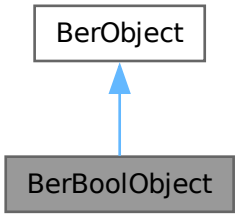
- inc/argument_helper_functions.h

4.3 BerBoolObject Class Reference

Inheritance diagram for BerBoolObject:



Collaboration diagram for BerBoolObject:



Public Member Functions

- berObjectTypes getBerObjectType ()

  Get type of BerObject.
- long long int getLenght ()

  Get the Lenght of BerObject representation in BER (including tag and lenght)
- std::vector< unsigned char > getBerRepresentation ()

  Returns the BER representation of BerObject.
- BerBoolObject (char value)

4.3.1 Detailed Description

Definition at line 15 of file BerBoolObject.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 BerBoolObject()

```
BerBoolObject::BerBoolObject (
            char value )
```

Definition at line 28 of file BerBoolObject.cpp.

4.3.2.2 ~BerBoolObject()

```
BerBoolObject::~BerBoolObject ( )
```

Definition at line 30 of file BerBoolObject.cpp.

4.3.3 Member Function Documentation

4.3.3.1 getBerObjectType()

```
berObjectTypes BerBoolObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented from BerObject.

Definition at line 8 of file BerBoolObject.cpp.

4.3.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerBoolObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

Returns

std::vector<unsigned char>

Reimplemented from BerObject.

Definition at line 15 of file BerBoolObject.cpp.

4.3.3.3 getLenght()

```
long long int BerBoolObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

Returns

long long int

Reimplemented from BerObject.

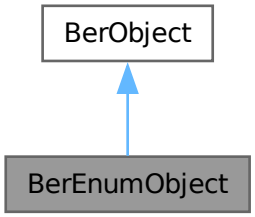
Definition at line 10 of file BerBoolObject.cpp.

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

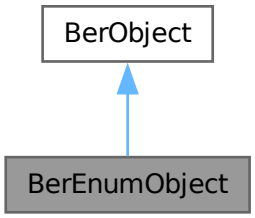
- inc/BerBoolObject.h
- src/BerBoolObject.cpp

4.4 BerEnumObject Class Reference

Inheritance diagram for BerEnumObject:



Collaboration diagram for BerEnumObject:



Public Member Functions

- berObjectTypes getBerObjectType ()

  Get type of BerObject.
- long long int getLenght ()

  Get the Lenght of BerObject representation in BER (including tag and lenght)
- std::vector< unsigned char > getBerRepresentation ()

  Returns the BER representation of BerObject.
- BerEnumObject (char value)

4.4.1 Detailed Description

Definition at line 15 of file BerEnumObject.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 BerEnumObject()

```
BerEnumObject::BerEnumObject (
            char value )
```

Definition at line 27 of file BerEnumObject.cpp.

4.4.2.2 ~BerEnumObject()

```
BerEnumObject::~BerEnumObject ( )
```

Definition at line 29 of file BerEnumObject.cpp.

4.4.3 Member Function Documentation

4.4.3.1 getBerObjectType()

```
berObjectTypes BerEnumObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented from BerObject.

Definition at line 10 of file BerEnumObject.cpp.

4.4.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerEnumObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

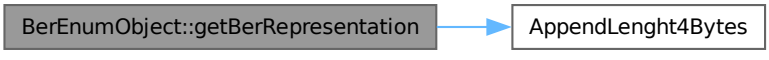
Returns

std::vector<unsigned char>

Reimplemented from BerObject.

Definition at line 18 of file BerEnumObject.cpp.

Here is the call graph for this function:



4.4.3.3 getLenght()

```
long long int BerEnumObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

Returns

long long int

Reimplemented from BerObject.

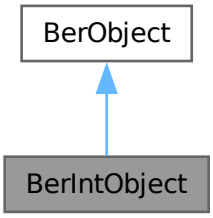
Definition at line 12 of file BerEnumObject.cpp.

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

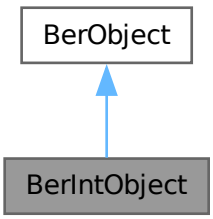
- inc/BerEnumObject.h
- src/BerEnumObject.cpp

4.5 BerIntObject Class Reference

Inheritance diagram for BerIntObject:



Collaboration diagram for BerIntObject:



Public Member Functions

- berObjectTypes getBerObjectType ()
  Get type of BerObject.
- int getValue ()
- void setValue (int value)
- long long int getLenght ()
  Get the Lenght of BerObject representation in BER (including tag and lenght)
- std::vector< unsigned char > getBerRepresentation ()
  Returns the BER representation of BerObject.
- BerIntObject (int value)

4.5.1 Detailed Description

Definition at line 15 of file BerIntObject.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 BerIntObject() [1/2]

```
BerIntObject::BerIntObject ( )
```

Definition at line 21 of file BerIntObject.cpp.

4.5.2.2 BerIntObject() [2/2]

```
BerIntObject::BerIntObject (
            int value )
```

Definition at line 22 of file BerIntObject.cpp.

4.5.2.3 ~BerIntObject()

```
BerIntObject::~BerIntObject ( )
```

Definition at line 28 of file BerIntObject.cpp.

4.5.3 Member Function Documentation

4.5.3.1 getBerObjectType()

```
berObjectTypes BerIntObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented from BerObject.

Definition at line 8 of file BerIntObject.cpp.

4.5.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerIntObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

Returns

std::vector<unsigned char>

Reimplemented from BerObject.

Definition at line 15 of file BerIntObject.cpp.

Here is the call graph for this function:

4.5.3.3 getLenght()

```
long long int BerIntObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

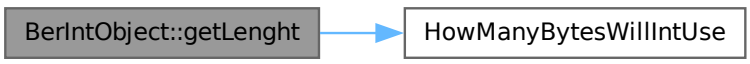
Returns

long long int

Reimplemented from BerObject.

Definition at line 10 of file BerIntObject.cpp.

Here is the call graph for this function:



4.5.3.4 getValue()

```
int BerIntObject::getValue ( )
```

Definition at line 24 of file BerIntObject.cpp.

4.5.3.5 setValue()

```
void BerIntObject::setValue (
            int value )
```

Definition at line 26 of file BerIntObject.cpp.

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

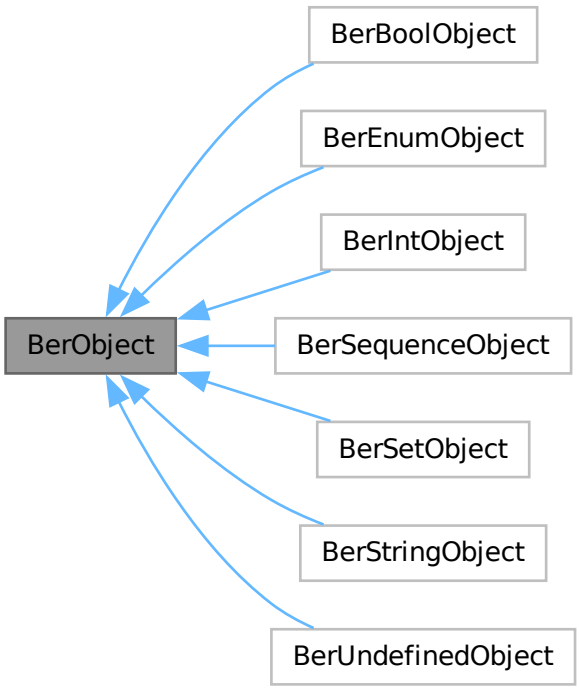
- inc/BerIntObject.h
- src/BerIntObject.cpp

4.6 BerObject Class Reference

Base class for all BER objects.

```
#include <BerObject.h>
```

Inheritance diagram for BerObject:



Public Member Functions

- virtual berObjectTypes getBerObjectType ()

  Get type of BerObject.
- virtual long long int getLenght ()

  Get the Lenght of BerObject representation in BER (including tag and lenght)
- virtual std::vector< unsigned char > getBerRepresentation ()

  Returns the BER representation of BerObject.

4.6.1 Detailed Description

Base class for all BER objects.

Definition at line 15 of file BerObject.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 ~BerObject()

```
BerObject::~BerObject ( )  [virtual]
```

Definition at line 20 of file BerObject.cpp.

4.6.3 Member Function Documentation

4.6.3.1 getBerObjectType()

```
berObjectTypes BerObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented in BerBoolObject, BerEnumObject, BerIntObject, BerSequenceObject, BerSetObject, BerStringObject, and BerUndefinedObject.

Definition at line 9 of file BerObject.cpp.

4.6.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

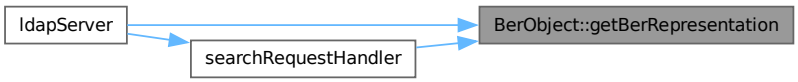
Returns

std::vector<unsigned char>

Reimplemented in BerBoolObject, BerEnumObject, BerIntObject, BerSequenceObject, BerSetObject, BerStringObject, and BerUndefinedObject.

Definition at line 15 of file BerObject.cpp.

Here is the caller graph for this function:



4.6.3.3 getLenght()

```
long long int BerObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

Returns

long long int

Reimplemented in BerBoolObject, BerEnumObject, BerIntObject, BerSequenceObject, BerSetObject, BerStringObject, and BerUndefinedObject.

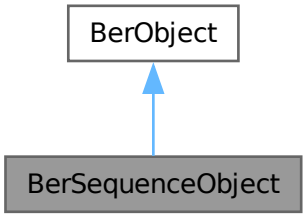
Definition at line 12 of file BerObject.cpp.

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

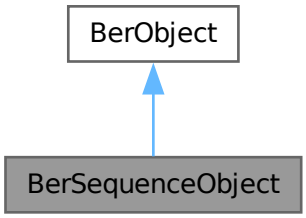
- inc/BerObject.h
- src/BerObject.cpp

4.7 BerSequenceObject Class Reference

Inheritance diagram for BerSequenceObject:



Collaboration diagram for BerSequenceObject:



Public Member Functions

- berObjectTypes getBerObjectType ()

  Get type of BerObject.
- long long int getLenght ()

  Get the Lenght of BerObject representation in BER (including tag and lenght)
- std::vector< unsigned char > getBerRepresentation ()

  Returns the BER representation of BerObject.
- BerSequenceObject (int tag)
- int GetTag ()

Public Attributes

- std::vector< BerObject * > objects

4.7.1 Detailed Description

Definition at line 16 of file BerSequenceObject.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 BerSequenceObject() [1/2]

```
BerSequenceObject::BerSequenceObject (
            int tag )
```

Definition at line 48 of file BerSequenceObject.cpp.

4.7.2.2 BerSequenceObject() [2/2]

```
BerSequenceObject::BerSequenceObject ( )
```

Definition at line 44 of file BerSequenceObject.cpp.

4.7.2.3 ~BerSequenceObject()

```
BerSequenceObject::~BerSequenceObject ( )
```

Definition at line 50 of file BerSequenceObject.cpp.

4.7.3 Member Function Documentation

4.7.3.1 getBerObjectType()

```
berObjectTypes BerSequenceObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented from BerObject.

Definition at line 8 of file BerSequenceObject.cpp.

4.7.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerSequenceObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

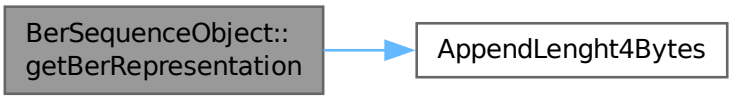
Returns

std::vector<unsigned char>

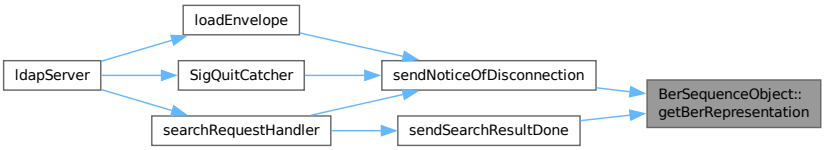
Reimplemented from BerObject.

Definition at line 24 of file BerSequenceObject.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



4.7.3.3 getLenght()

```
long long int BerSequenceObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

Returns

long long int

Reimplemented from BerObject.

Definition at line 12 of file BerSequenceObject.cpp.

4.7.3.4 GetTag()

```
int BerSequenceObject::GetTag ( )
```

Definition at line 46 of file BerSequenceObject.cpp.

4.7.4 Member Data Documentation

4.7.4.1 objects

```
std::vector<BerObject *> BerSequenceObject::objects
```

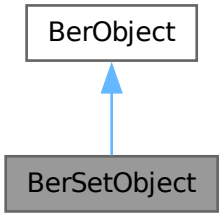
Definition at line 21 of file BerSequenceObject.h.

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

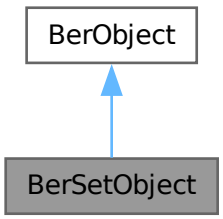
- inc/BerSequenceObject.h
- src/BerSequenceObject.cpp

4.8 BerSetObject Class Reference

Inheritance diagram for BerSetObject:



Collaboration diagram for BerSetObject:



Public Member Functions

- berObjectTypes getBerObjectType ()

  Get type of BerObject.
- long long int getLenght ()

  Get the Lenght of BerObject representation in BER (including tag and lenght)
- std::vector< unsigned char > getBerRepresentation ()

  Returns the BER representation of BerObject.

Public Attributes

- std::vector< BerObject * > objects

4.8.1 Detailed Description

Definition at line 16 of file BerSetObject.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 BerSetObject()

```
BerSetObject::BerSetObject ( )
```

Definition at line 42 of file BerSetObject.cpp.

4.8.2.2 ~BerSetObject()

```
BerSetObject::~BerSetObject ( )
```

Definition at line 44 of file BerSetObject.cpp.

4.8.3 Member Function Documentation

4.8.3.1 getBerObjectType()

```
berObjectTypes BerSetObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented from BerObject.

Definition at line 8 of file BerSetObject.cpp.

4.8.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerSetObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

Returns

std::vector<unsigned char>

Reimplemented from BerObject.

Definition at line 22 of file BerSetObject.cpp.

Here is the call graph for this function:

4.8.3.3 getLenght()

```
long long int BerSetObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

Returns

long long int

Reimplemented from BerObject.

Definition at line 10 of file BerSetObject.cpp.

4.8.4 Member Data Documentation

4.8.4.1 objects

```
std::vector<BerObject *> BerSetObject::objects
```

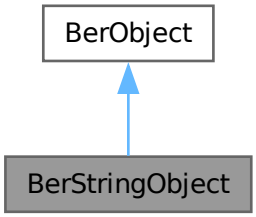
Definition at line 18 of file BerSetObject.h.

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

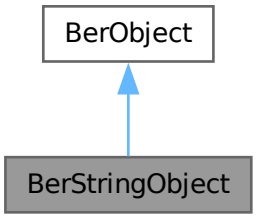
- inc/BerSetObject.h
- src/BerSetObject.cpp

4.9 BerStringObject Class Reference

Inheritance diagram for BerStringObject:



Collaboration diagram for BerStringObject:



Public Member Functions

- berObjectTypes getBerObjectType ()

  Get type of BerObject.
- long long int getLenght ()

  Get the Lenght of BerObject representation in BER (including tag and lenght)
- std::vector< unsigned char > getBerRepresentation ()

  Returns the BER representation of BerObject.
- BerStringObject (std::vector< unsigned char > value)
- BerStringObject (std::string value)

Public Attributes

- std::vector< unsigned char > value

4.9.1 Detailed Description

Definition at line 15 of file BerStringObject.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 BerStringObject() [1/3]

```
BerStringObject::BerStringObject ( )
```

Definition at line 23 of file BerStringObject.cpp.

4.9.2.2 BerStringObject() [2/3]

```
BerStringObject::BerStringObject (
            std::vector< unsigned char > value )
```

Definition at line 25 of file BerStringObject.cpp.

4.9.2.3 BerStringObject() [3/3]

```
BerStringObject::BerStringObject (
            std::string value )
```

Definition at line 28 of file BerStringObject.cpp.

4.9.3 Member Function Documentation

4.9.3.1 getBerObjectType()

```
berObjectTypes BerStringObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented from BerObject.

Definition at line 8 of file BerStringObject.cpp.

4.9.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerStringObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

Returns

std::vector<unsigned char>

Reimplemented from BerObject.

Definition at line 15 of file BerStringObject.cpp.

Here is the call graph for this function:

4.9.3.3 getLenght()

```
long long int BerStringObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

Returns

long long int

Reimplemented from BerObject.

Definition at line 9 of file BerStringObject.cpp.

4.9.4 Member Data Documentation

4.9.4.1 value

```
std::vector<unsigned char> BerStringObject::value
```

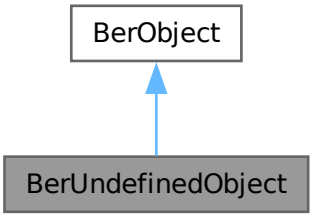
Definition at line 18 of file BerStringObject.h.

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

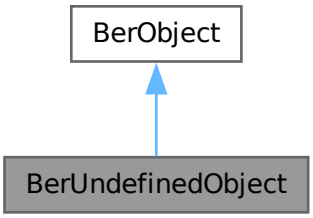
- inc/BerStringObject.h
- src/BerStringObject.cpp

4.10 BerUndefinedObject Class Reference

Inheritance diagram for BerUndefinedObject:



Collaboration diagram for BerUndefinedObject:



Public Member Functions

- berObjectTypes getBerObjectType ()

  Get type of BerObject.
- long long int getLenght ()

  Get the Lenght of BerObject representation in BER (including tag and lenght)
- std::vector< unsigned char > getBerRepresentation ()

  Returns the BER representation of BerObject.
- BerUndefinedObject (std::vector< unsigned char > value)

4.10.1 Detailed Description

Definition at line 12 of file BerUndefinedObject.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 BerUndefinedObject()

```
BerUndefinedObject::BerUndefinedObject (
            std::vector< unsigned char > value )
```

Definition at line 20 of file BerUndefinedObject.cpp.

4.10.3 Member Function Documentation

4.10.3.1 getBerObjectType()

```
berObjectTypes BerUndefinedObject::getBerObjectType ( )  [virtual]
```

Get type of BerObject.

Returns

berObjectTypes

Reimplemented from BerObject.

Definition at line 9 of file BerUndefinedObject.cpp.

4.10.3.2 getBerRepresentation()

```
std::vector< unsigned char > BerUndefinedObject::getBerRepresentation ( )  [virtual]
```

Returns the BER representation of BerObject.

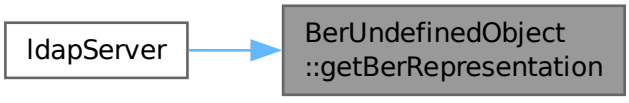
Returns

std::vector<unsigned char>

Reimplemented from BerObject.

Definition at line 16 of file BerUndefinedObject.cpp.

Here is the caller graph for this function:



4.10.3.3 getLenght()

```
long long int BerUndefinedObject::getLenght ( )  [virtual]
```

Get the Lenght of BerObject representation in BER (including tag and lenght)

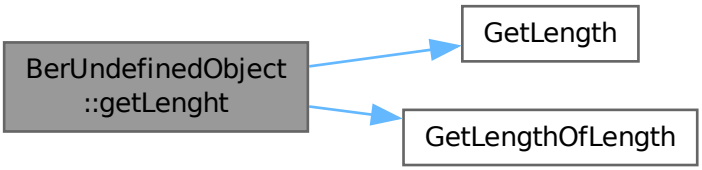
Returns

long long int

Reimplemented from BerObject.

Definition at line 12 of file BerUndefinedObject.cpp.

Here is the call graph for this function:



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

- inc/BerUndefinedObject.h
- src/BerUndefinedObject.cpp

4.11 DatabaseController Class Reference

class for loading and parsing database file

```
#include <DatabaseController.h>
```

Public Member Functions

- DatabaseObject loadNextRow (int *err)
  loads next row from database file
- std::vector< DatabaseObject > loadAllRows ()
  loads all rows from database file
- DatabaseController (std::string fileName)
  Construct a new Database Controller object.

4.11.1 Detailed Description

class for loading and parsing database file

Definition at line 20 of file DatabaseController.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 DatabaseController()

```
DatabaseController::DatabaseController (
            std::string fileName )
```

Construct a new Database Controller object.

Parameters

fileName	path to database csv file

Definition at line 67 of file DatabaseController.cpp.

4.11.2.2 ~DatabaseController()

```
DatabaseController::~DatabaseController ( )
```

Definition at line 71 of file DatabaseController.cpp.

4.11.3 Member Function Documentation

4.11.3.1 loadAllRows()

```
std::vector< DatabaseObject > DatabaseController::loadAllRows ( )
```

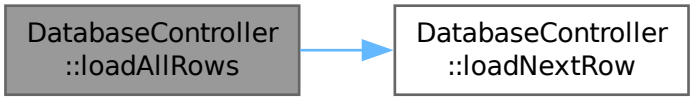
loads all rows from database file

Returns

std::vector<DatabaseObject>

Definition at line 8 of file DatabaseController.cpp.

Here is the call graph for this function:



4.11.3.2 loadNextRow()

```
DatabaseObject DatabaseController::loadNextRow (
            int * err )
```

loads next row from database file

Parameters

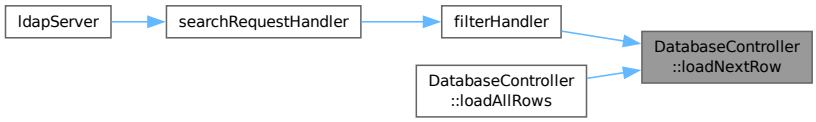
err	1 if EOF, 0 if success

Returns

DatabaseObject

Definition at line 29 of file DatabaseController.cpp.

Here is the caller graph for this function:



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

- inc/DatabaseController.h
- src/DatabaseController.cpp

4.12 DatabaseObject Class Reference

Object representing one row from database.

```
#include <DatabaseObject.h>
```

Public Member Functions

- std::vector< unsigned char > get_name ()
- std::vector< unsigned char > get_uid ()
- std::vector< unsigned char > get_email ()
- DatabaseObject (std::vector< unsigned char > name, std::vector< unsigned char > uid, std::vector< unsigned char > email)

4.12.1 Detailed Description

Object representing one row from database.

Definition at line 19 of file DatabaseObject.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 DatabaseObject()

```
DatabaseObject::DatabaseObject (
            std::vector< unsigned char > name,
            std::vector< unsigned char > uid,
            std::vector< unsigned char > email )
```

Definition at line 11 of file DatabaseObject.cpp.

4.12.3 Member Function Documentation

4.12.3.1 get_email()

```
std::vector< unsigned char > DatabaseObject::get_email ( )
```

Definition at line 10 of file DatabaseObject.cpp.

4.12.3.2 get_name()

```
std::vector< unsigned char > DatabaseObject::get_name ( )
```

Definition at line 8 of file DatabaseObject.cpp.

4.12.3.3 get_uid()

```
std::vector< unsigned char > DatabaseObject::get_uid ( )
```

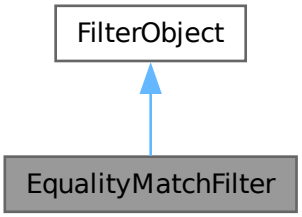
Definition at line 9 of file DatabaseObject.cpp.

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

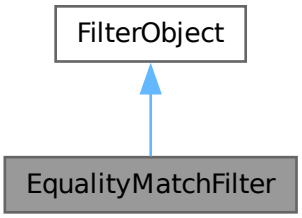
- inc/DatabaseObject.h
- src/DatabaseObject.cpp

4.13 EqualityMatchFilter Class Reference

Inheritance diagram for EqualityMatchFilter:



Collaboration diagram for EqualityMatchFilter:



Public Member Functions

- EqualityMatchFilter (std::vector< unsigned char > attributeDescription, std::vector< unsigned char > assertionValue)
- std::vector< unsigned char > getAttributeDescription ()
- std::vector< unsigned char > getAssertionValue ()
- filterTypes getFilterType ()

4.13.1 Detailed Description

Definition at line 15 of file EqualityMatchFilterObject.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 EqualityMatchFilter()

```
EqualityMatchFilter::EqualityMatchFilter (
            std::vector< unsigned char > attributeDescription,
            std::vector< unsigned char > assertionValue )
```

Definition at line 8 of file EqualityMatchFilterObject.cpp.

4.13.3 Member Function Documentation

4.13.3.1 getAssertionValue()

```
std::vector< unsigned char > EqualityMatchFilter::getAssertionValue ( )
```

Definition at line 19 of file EqualityMatchFilterObject.cpp.

4.13.3.2 getAttributeDescription()

```
std::vector< unsigned char > EqualityMatchFilter::getAttributeDescription ( )
```

Definition at line 15 of file EqualityMatchFilterObject.cpp.

4.13.3.3 getFilterType()

```
filterTypes EqualityMatchFilter::getFilterType ( )  [virtual]
```

Reimplemented from FilterObject.

Definition at line 23 of file EqualityMatchFilterObject.cpp.

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

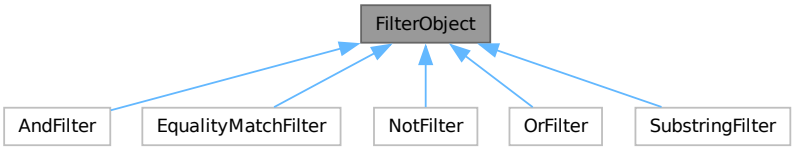
- inc/EqualityMatchFilterObject.h
- src/EqualityMatchFilterObject.cpp

4.14 FilterObject Class Reference

base class for all filter objects

```
#include <FilterObject.h>
```

Inheritance diagram for FilterObject:



Public Member Functions

- virtual filterTypes getFilterType ()

4.14.1 Detailed Description

base class for all filter objects

Definition at line 17 of file FilterObject.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 ~FilterObject()

```
FilterObject::~FilterObject ( )  [virtual]
```

Definition at line 10 of file FilterObject.cpp.

4.14.3 Member Function Documentation

4.14.3.1 getFilterType()

```
filterTypes FilterObject::getFilterType ( )  [virtual]
```

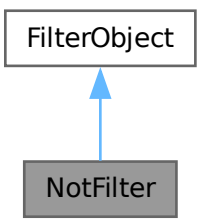
Definition at line 8 of file FilterObject.cpp.

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

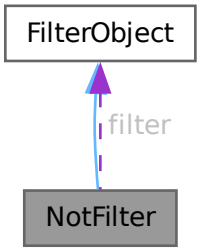
- inc/FilterObject.h
- src/FilterObject.cpp

4.15 NotFilter Class Reference

Inheritance diagram for NotFilter:



Collaboration diagram for NotFilter:



Public Member Functions

- filterTypes getFilterType ()

Public Attributes

- FilterObject * filter

4.15.1 Detailed Description

Definition at line 14 of file NotFilterObject.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 ~NotFilter()

```
NotFilter::~NotFilter ( )
```

Definition at line 10 of file NotFilterObject.cpp.

4.15.3 Member Function Documentation

4.15.3.1 getFilterType()

```
filterTypes NotFilter::getFilterType ( )  [virtual]
```

Reimplemented from FilterObject.

Definition at line 8 of file NotFilterObject.cpp.

4.15.4 Member Data Documentation

4.15.4.1 filter

```
FilterObject* NotFilter::filter
```

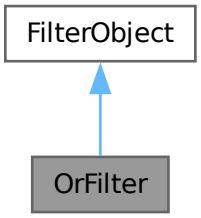
Definition at line 16 of file NotFilterObject.h.

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

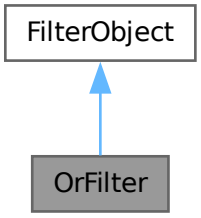
- inc/NotFilterObject.h
- src/NotFilterObject.cpp

4.16 OrFilter Class Reference

Inheritance diagram for OrFilter:



Collaboration diagram for OrFilter:



Public Member Functions

- filterTypes getFilterType ()

Public Attributes

- std::vector< FilterObject * > filters

4.16.1 Detailed Description

Definition at line 14 of file OrFilterObject.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 ~OrFilter()

```
OrFilter::~OrFilter ( )
```

Definition at line 10 of file OrFilterObject.cpp.

4.16.3 Member Function Documentation

4.16.3.1 getFilterType()

```
filterTypes OrFilter::getFilterType ( )  [virtual]
```

Reimplemented from FilterObject.

Definition at line 8 of file OrFilterObject.cpp.

4.16.4 Member Data Documentation

4.16.4.1 filters

```
std::vector<FilterObject *> OrFilter::filters
```

Definition at line 16 of file OrFilterObject.h.

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

- inc/OrFilterObject.h
- src/OrFilterObject.cpp

4.17 searchedAttributes Struct Reference

Public Attributes

- bool cn
- bool email
- bool uid

4.17.1 Detailed Description

Definition at line 45 of file ldap_comunication.h.

4.17.2 Member Data Documentation

4.17.2.1 cn

```
bool searchedAttributes::cn
```

Definition at line 46 of file ldap_comunication.h.

4.17.2.2 email

```
bool searchedAttributes::email
```

Definition at line 47 of file ldap_comunication.h.

4.17.2.3 uid

```
bool searchedAttributes::uid
```

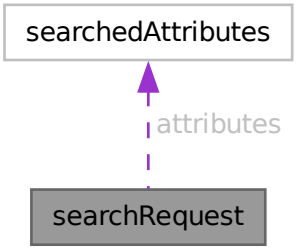
Definition at line 48 of file ldap_comunication.h.

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

- inc/ldap_comunication.h

4.18 searchRequest Struct Reference

Collaboration diagram for searchRequest:



Public Attributes

- int messageIDLength
- unsigned int sizeLimit
- searchedAttributesType attributes

4.18.1 Detailed Description

Definition at line 66 of file ldap_comunication.h.

4.18.2 Member Data Documentation

4.18.2.1 attributes

```
searchedAttributesType searchRequest::attributes
```

Definition at line 69 of file ldap_comunication.h.

4.18.2.2 messageIDLength

```
int searchRequest::messageIDLength
```

Definition at line 67 of file ldap_comunication.h.

4.18.2.3 sizeLimit

```
unsigned int searchRequest::sizeLimit
```

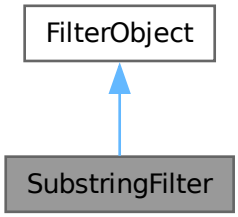
Definition at line 68 of file ldap_comunication.h.

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

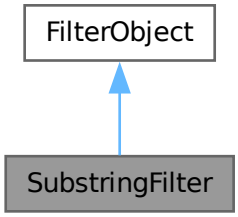
- inc/ldap_comunication.h

4.19 SubstringFilter Class Reference

Inheritance diagram for SubstringFilter:



Collaboration diagram for SubstringFilter:



Public Member Functions

- SubstringFilter (std::vector< unsigned char > attributeDescription, std::vector< unsigned char > subInitial, std::vector< std::vector< unsigned char > > subAny, std::vector< unsigned char > subFinal)
- std::vector< unsigned char > getAttributeDescription ()
- std::vector< unsigned char > getSubInitial ()
- std::vector< std::vector< unsigned char > > getSubAny ()
- std::vector< unsigned char > getSubFinal ()
- filterTypes getFilterType ()

4.19.1 Detailed Description

Definition at line 13 of file SubstringFilterObject.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 SubstringFilter()

```
SubstringFilter::SubstringFilter (
            std::vector< unsigned char > attributeDescription,
            std::vector< unsigned char > subInitial,
            std::vector< std::vector< unsigned char > > subAny,
            std::vector< unsigned char > subFinal )
```

Definition at line 8 of file SubstringFilterObject.cpp.

4.19.3 Member Function Documentation

4.19.3.1 getAttributeDescription()

```
std::vector< unsigned char > SubstringFilter::getAttributeDescription ( )
```

Definition at line 19 of file SubstringFilterObject.cpp.

4.19.3.2 getFilterType()

```
filterTypes SubstringFilter::getFilterType ( )  [virtual]
```

Reimplemented from FilterObject.

Definition at line 29 of file SubstringFilterObject.cpp.

4.19.3.3 getSubAny()

```
std::vector< std::vector< unsigned char > > SubstringFilter::getSubAny ( )
```

Definition at line 25 of file SubstringFilterObject.cpp.

4.19.3.4 getSubFinal()

```
std::vector< unsigned char > SubstringFilter::getSubFinal ( )
```

Definition at line 28 of file SubstringFilterObject.cpp.

4.19.3.5 getSubInitial()

```
std::vector< unsigned char > SubstringFilter::getSubInitial ( )
```

Definition at line 22 of file SubstringFilterObject.cpp.

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

- inc/SubstringFilterObject.h
- src/SubstringFilterObject.cpp

Chapter 5

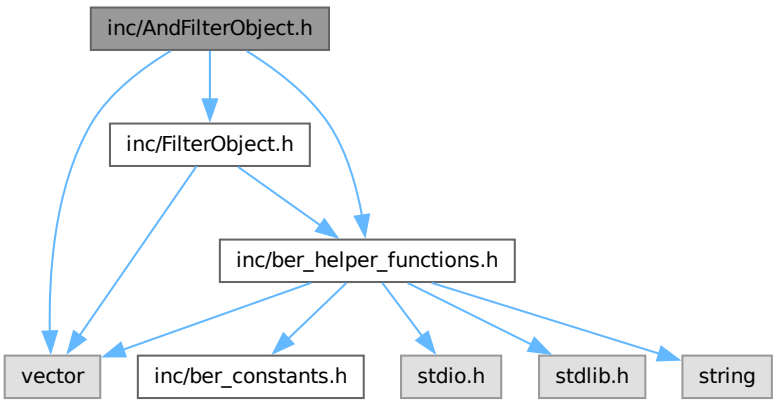
File Documentation

5.1 inc/AndFilterObject.h File Reference

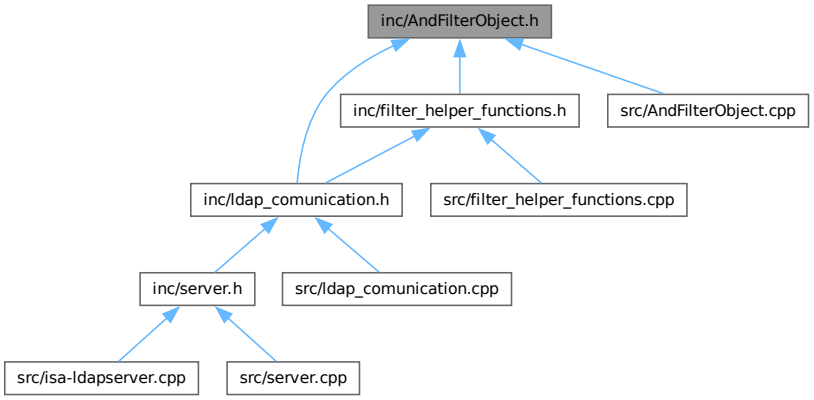
And filter object for BER LDAP.

```
#include "inc/FilterObject.h"
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for AndFilterObject.h:



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



Classes

- class AndFilter

5.1.1 Detailed Description

And filter object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file AndFilterObject.h.

5.2 AndFilterObject.h

Go to the documentation of this file.

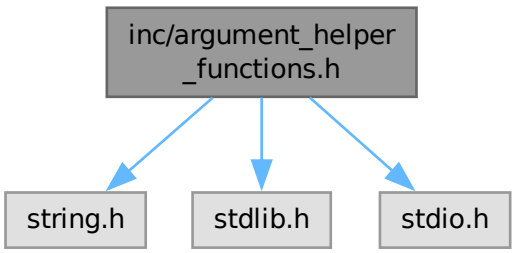
```
00001
00007 #ifndef ANDFILTEROBJECT_H
00008 #define ANDFILTEROBJECT_H
00009 #include "inc/FilterObject.h"
00010 #include "inc/ber_helper_functions.h"
00011
00012 #include <vector>
00013
00014 class AndFilter : public FilterObject {
00015 public:
00016   std::vector<FilterObject *> filters;
00017   filterTypes getFilterType();
00018   ~AndFilter();
00019 };
00020
00021 #endif
```

5.3 inc/argument_helper_functions.h File Reference

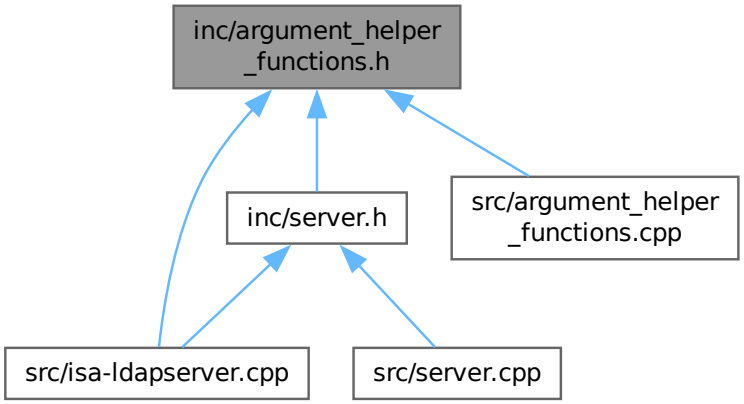
Helper functions for parsing arguments.

```
#include "string.h"
#include <stdlib.h>
#include <stdio.h>
```

Include dependency graph for argument_helper_functions.h:



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



Classes

- struct args_t

Typedefs

- typedef struct args_t **argsT**

Functions

- argsT parseArguments (int argc, const char **argv)

  Parses the arguments from the command line for ldapserver.

5.3.1 Detailed Description

Helper functions for parsing arguments.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file argument_helper_functions.h.

5.3.2 Function Documentation

5.3.2.1 parseArguments()

```
argsT parseArguments (
            int argc,
            const char ** argv )
```

Parses the arguments from the command line for ldapserver.

Parameters

argc	count of arguments
argv	values of arguments

Returns

argsT

Definition at line 8 of file argument_helper_functions.cpp.

5.4 argument_helper_functions.h

Go to the documentation of this file.

```
00001
00007 #ifndef ARGUMENT_HELPER_FUNCTIONS_H
00008 #define ARGUMENT_HELPER_FUNCTIONS_H
00009 #include "string.h"
00010 #include <stdlib.h>
```

```
00011 #include <stdio.h>
00012
00013 typedef struct args_t {
00014   char *dbPath;
00015   int port;
00016   bool err;
00017 } argsT;
00018
00026 argsT parseArguments(int argc,const char **argv);
00027
00028 #endif
```

5.5 inc/ber_constants.h File Reference

Constants that are used in LDAP BER.

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

Variables

- const unsigned int BER_TAG_LENGTH = 1
- const unsigned int BER_LENGTH_OF_LENGTH_TAG = 1
- const unsigned int BER_4BYTE_LENGTH_LENGTH = 4
- const unsigned int BER_EXTENDED_RESPONSE_C = 0x78
- const unsigned int BER_BIND_REQUEST_C = 0x60
- const unsigned int BER_BIND_RESPONSE_C = 0x61
- const unsigned int BER_SEARCH_REQUEST_C = 0x63
- const unsigned int BER_SEARCH_RESULT_ENTRY_C = 0x64
- const unsigned int BER_SEARCH_RESULT_DONE_C = 0x65
- const unsigned int BER_UNBIND_REQUEST_C = 0x42
- const unsigned int BER_BOOL_C = 0x01
- const unsigned int BER_INT_C = 0x02
- const unsigned int BER_INT_4BYTES_C = 0x84
- const unsigned int BER_OCTET_STRING_C = 0x04
- const unsigned int BER_ENUM_C = 0x0A
- const unsigned int BER_SEQUENCE_C = 0x30
- const unsigned int BER_SET_C = 0x31
- const unsigned int BER_LDAP_SUCCESS = 0x00
- const unsigned int BER_LDAP_PROTOCOL_ERROR = 0x02
- const unsigned int BER_LDAP_UNAVAILABLE = 0x34
- const unsigned int BER_LDAP_SIZE_LIMIT_EXCEEDED = 0x04
- const unsigned int BER_LDAP_AUTH_METHOD_NOT_SUPPORTED = 0x07

5.5.1 Detailed Description

Constants that are used in LDAP BER.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file ber_constants.h.

5.5.2 Variable Documentation

5.5.2.1 BER_4BYTE_LENGTH_LENGTH

```
const unsigned int BER_4BYTE_LENGTH_LENGTH = 4
```

Definition at line 12 of file ber_constants.h.

5.5.2.2 BER_BIND_REQUEST_C

```
const unsigned int BER_BIND_REQUEST_C = 0x60
```

Definition at line 14 of file ber_constants.h.

5.5.2.3 BER_BIND_RESPONSE_C

```
const unsigned int BER_BIND_RESPONSE_C = 0x61
```

Definition at line 15 of file ber_constants.h.

5.5.2.4 BER_BOOL_C

```
const unsigned int BER_BOOL_C = 0x01
```

Definition at line 21 of file ber_constants.h.

5.5.2.5 BER_ENUM_C

```
const unsigned int BER_ENUM_C = 0x0A
```

Definition at line 25 of file ber_constants.h.

5.5.2.6 BER_EXTENDED_RESPONSE_C

```
const unsigned int BER_EXTENDED_RESPONSE_C = 0x78
```

Definition at line 13 of file ber_constants.h.

5.5.2.7 BER_INT_4BYTES_C

```
const unsigned int BER_INT_4BYTES_C = 0x84
```

Definition at line 23 of file ber_constants.h.

5.5.2.8 BER_INT_C

```
const unsigned int BER_INT_C = 0x02
```

Definition at line 22 of file ber_constants.h.

5.5.2.9 BER_LDAP_AUTH_METHOD_NOT_SUPPORTED

```
const unsigned int BER_LDAP_AUTH_METHOD_NOT_SUPPORTED = 0x07
```

Definition at line 35 of file ber_constants.h.

5.5.2.10 BER_LDAP_PROTOCOL_ERROR

```
const unsigned int BER_LDAP_PROTOCOL_ERROR = 0x02
```

Definition at line 32 of file ber_constants.h.

5.5.2.11 BER_LDAP_SIZE_LIMIT_EXCEEDED

```
const unsigned int BER_LDAP_SIZE_LIMIT_EXCEEDED = 0x04
```

Definition at line 34 of file ber_constants.h.

5.5.2.12 BER_LDAP_SUCCESS

```
const unsigned int BER_LDAP_SUCCESS = 0x00
```

Definition at line 31 of file ber_constants.h.

5.5.2.13 BER_LDAP_UNAVAILABLE

```
const unsigned int BER_LDAP_UNAVAILABLE = 0x34
```

Definition at line 33 of file ber_constants.h.

5.5.2.14 BER_LENGTH_OF_LENGTH_TAG

```
const unsigned int BER_LENGTH_OF_LENGTH_TAG = 1
```

Definition at line 11 of file ber_constants.h.

5.5.2.15 BER_OCTET_STRING_C

```
const unsigned int BER_OCTET_STRING_C = 0x04
```

Definition at line 24 of file ber_constants.h.

5.5.2.16 BER_SEARCH_REQUEST_C

```
const unsigned int BER_SEARCH_REQUEST_C = 0x63
```

Definition at line 16 of file ber_constants.h.

5.5.2.17 BER_SEARCH_RESULT_DONE_C

```
const unsigned int BER_SEARCH_RESULT_DONE_C = 0x65
```

Definition at line 18 of file ber_constants.h.

5.5.2.18 BER_SEARCH_RESULT_ENTRY_C

```
const unsigned int BER_SEARCH_RESULT_ENTRY_C = 0x64
```

Definition at line 17 of file ber_constants.h.

5.5.2.19 BER_SEQUENCE_C

```
const unsigned int BER_SEQUENCE_C = 0x30
```

Definition at line 26 of file ber_constants.h.

5.5.2.20 BER_SET_C

```
const unsigned int BER_SET_C = 0x31
```

Definition at line 27 of file ber_constants.h.

5.5.2.21 BER_TAG_LENGTH

```
const unsigned int BER_TAG_LENGTH = 1
```

Definition at line 10 of file ber_constants.h.

5.5.2.22 BER_UNBIND_REQUEST_C

```
const unsigned int BER_UNBIND_REQUEST_C = 0x42
```

Definition at line 19 of file ber_constants.h.

5.6 ber_constants.h

Go to the documentation of this file.

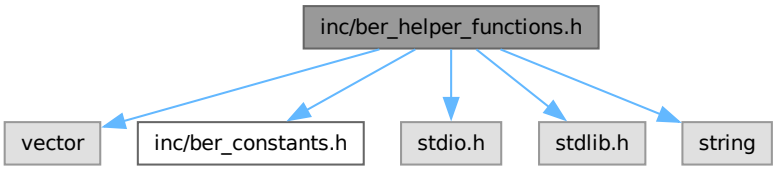
```
00001
00008 #ifndef BER_CONSTANTS_H
00009 #define BER_CONSTANTS_H
00010 const unsigned int BER_TAG_LENGTH = 1;
00011 const unsigned int BER_LENGTH_OF_LENGTH_TAG = 1;
00012 const unsigned int BER_4BYTE_LENGTH_LENGTH = 4;
00013 const unsigned int BER_EXTENDED_RESPONSE_C = 0x78;
00014 const unsigned int BER_BIND_REQUEST_C = 0x60;
00015 const unsigned int BER_BIND_RESPONSE_C = 0x61;
00016 const unsigned int BER_SEARCH_REQUEST_C = 0x63;
00017 const unsigned int BER_SEARCH_RESULT_ENTRY_C = 0x64;
00018 const unsigned int BER_SEARCH_RESULT_DONE_C = 0x65;
00019 const unsigned int BER_UNBIND_REQUEST_C = 0x42;
00020
00021 const unsigned int BER_BOOL_C = 0x01;
00022 const unsigned int BER_INT_C = 0x02;
00023 const unsigned int BER_INT_4BYTES_C = 0x84;
00024 const unsigned int BER_OCTET_STRING_C = 0x04;
00025 const unsigned int BER_ENUM_C = 0x0A;
00026 const unsigned int BER_SEQUENCE_C = 0x30;
00027 const unsigned int BER_SET_C = 0x31;
00028
00029 //constants --- result codes
00030
00031 const unsigned int BER_LDAP_SUCCESS = 0x00;
00032 const unsigned int BER_LDAP_PROTOCOL_ERROR = 0x02;
00033 const unsigned int BER_LDAP_UNAVAILABLE = 0x34;
00034 const unsigned int BER_LDAP_SIZE_LIMIT_EXCEEDED = 0x04;
00035 const unsigned int BER_LDAP_AUTH_METHOD_NOT_SUPPORTED = 0x07;
00036
00037
00038
00039 #endif
```

5.7 inc/ber_helper_functions.h File Reference

Helper functions for parsing BER LDAP.

```
#include <vector>
#include "inc/ber_constants.h"
#include <stdio.h>
#include <stdlib.h>
#include <string>
```

Include dependency graph for ber_helper_functions.h:



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

Typedefs

- typedef enum filterTypes filterTypes

  writes int in BER LDAP format to char array
- typedef enum berObjectTypes **berObjectTypes**

Enumerations

- enum filterTypes {
  AND , **OR** , **NOT** , **equalityMatch** ,
  substrings , **undefined** }

  writes int in BER LDAP format to char array
- enum **berObjectTypes** {
  berSequenceObject , **berIntObject** , **berStringObject** , **berSetObject** ,
  berEnumObject , **berBoolObject** , **berUndefined** , **berErr** }

Functions

- std::vector< unsigned char > ToLowerCase (std::vector< unsigned char > input)

  converts std::vector<unsigned char> to lowercase
- int ParseINT (unsigned char *s, int *err)

  parses 1 integer from ldap coded message
- int HowManyBytesWillIntUse (int value)

  returns the number of bytes that will be used to encode the int
- int WriteIntAppend (std::vector< unsigned char > &s, int value)

  writes int in BER LDAP format to char array
- void AppendLenght4Bytes (std::vector< unsigned char > &start, int value)

  appends length in BER LDAP format to char array (4 bytes)
- int GetLength (std::vector< unsigned char >::iterator start, int *err, std::vector< unsigned char >::iterator end)

  Get the length of the data of BER attribute.
- unsigned int ParseINT (std::vector< unsigned char >::iterator s, int *err, std::vector< unsigned char >↩ ::iterator end)

  Parses value of BERInteger from char array.
- int GetLengthOfLength (std::vector< unsigned char >::iterator start, int *err, std::vector< unsigned char >↩ ::iterator end)

  Get the Length Of Length of BER attribute.
- void SkipTags (std::vector< unsigned char >::iterator &start, int n, int *err, std::vector< unsigned char >↩ ::iterator end)

  skips n BER attributes from char array, returns incremented iterator
- void GoIntoTag (std::vector< unsigned char >::iterator &start, int *err, std::vector< unsigned char >::iterator end)

  goes into the sequence/set tag and returns incremented iterator which points to the first attribute in the sequence/set
- void IncreaseLength4Bytes (std::vector< unsigned char >::iterator &start, int n, int *err, std::vector< unsigned char >::iterator end)

  Increases 4Bytes longform length of the attribute by n.
- filterTypes getFilterType (std::vector< unsigned char >::iterator start)

  Parses type of filter from char array and returns its enum.

5.7.1 Detailed Description

Helper functions for parsing BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file ber_helper_functions.h.

5.7.2 Typedef Documentation

5.7.2.1 filterTypes

```
typedef enum filterTypes filterTypes
```

writes int in BER LDAP format to char array

Parameters

s	start of the string in char array
value	int to be written

Returns

-1 if error, 0 if success

5.7.3 Enumeration Type Documentation

5.7.3.1 berObjectTypes

```
enum berObjectTypes
```

Definition at line 47 of file ber_helper_functions.h.

5.7.3.2 filterTypes

```
enum filterTypes
```

writes int in BER LDAP format to char array

Parameters

s	start of the string in char array
value	int to be written

Returns

-1 if error, 0 if success

Definition at line 38 of file ber_helper_functions.h.

5.7.4 Function Documentation

5.7.4.1 AppendLenght4Bytes()

```
void AppendLenght4Bytes (
            std::vector< unsigned char > & start,
            int value )
```

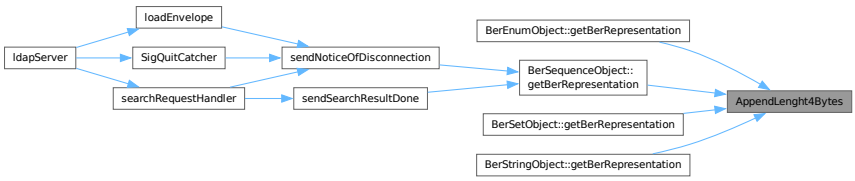
appends length in BER LDAP format to char array (4 bytes)

Parameters

start	vector to append to
value	
err	

Definition at line 161 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.7.4.2 getFilterType()

```
filterTypes getFilterType (
            std::vector< unsigned char >::iterator start )
```

Parses type of filter from char array and returns its enum.

Parameters

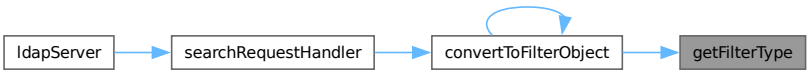
start	start of the filter

Returns

filterTypes

Definition at line 8 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.7.4.3 GetLength()

```
int GetLength (
            std::vector< unsigned char >::iterator start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Get the length of the data of BER attribute.

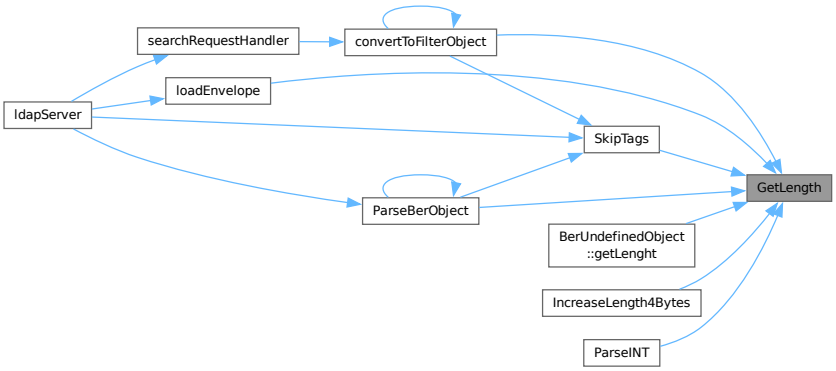
Parameters

start	start of the length
err	1 if error, 0 if success
end	end of the array

Returns

Definition at line 90 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.7.4.4 GetLengthOfLength()

```
int GetLengthOfLength (
            std::vector< unsigned char >::iterator start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Get the Length Of Length of BER attribute.

Parameters

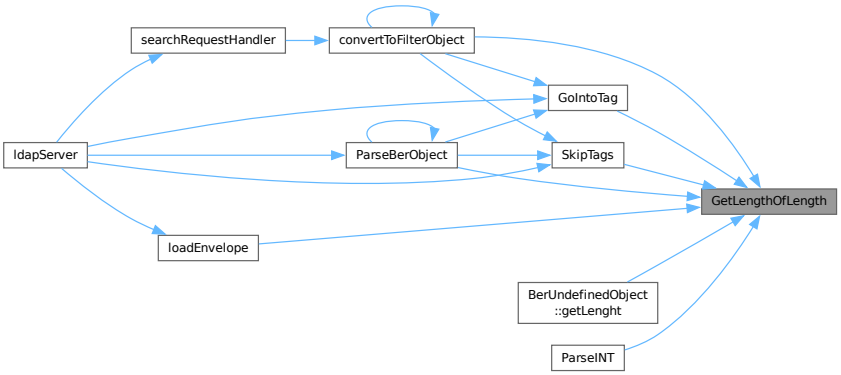
start	start of the length
err	1 if error, 0 if success
end	end of the array

Returns

int

Definition at line 63 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.7.4.5 GoIntoTag()

```
void GoIntoTag (
            std::vector< unsigned char >::iterator & start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

goes into the sequence/set tag and returns incremented iterator which points to the first attribute in the sequence/set

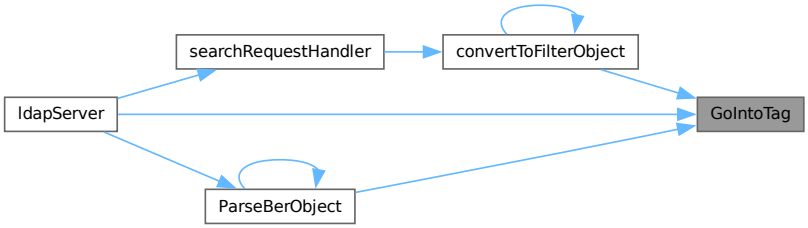
Parameters

start	start of the tag
err	1 if error, 0 if success
end	end of the array

Definition at line 254 of file ber_helper_functions.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



5.7.4.6 HowManyBytesWillIntUse()

```
int HowManyBytesWillIntUse (
            int value )
```

returns the number of bytes that will be used to encode the int

Parameters

value	

Returns

int

Definition at line 219 of file ber_helper_functions.cpp.

Here is the caller graph for this function:

5.7.4.7 IncreaseLength4Bytes()

```
void IncreaseLength4Bytes (
            std::vector< unsigned char >::iterator & start,
            int n,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Increases 4Bytes longform length of the attribute by n.

Parameters

start	start of the length
n	number by which the length will be increased
err	1 if error, 0 if success
end	end of the array

Definition at line 150 of file ber_helper_functions.cpp.

Here is the call graph for this function:

5.7.4.8 ParseINT() [1/2]

```
unsigned int ParseINT (
            std::vector< unsigned char >::iterator s,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Parses value of BERInteger from char array.

Parameters

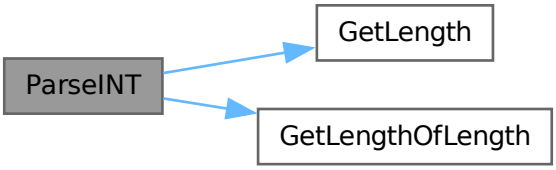
s	start of the integer in char array
err	1 if error, 0 if success
end	end of the array

Returns

unsigned int

Definition at line 23 of file ber_helper_functions.cpp.

Here is the call graph for this function:



5.7.4.9 ParseINT() [2/2]

```
int ParseINT (
            unsigned char * s,
            int * err )
```

parses 1 integer from ldap coded message

Parameters

s	start of the integer in char array

Returns

int - parsed integer

Here is the caller graph for this function:

5.7.4.10 SkipTags()

```
void SkipTags (
            std::vector< unsigned char >::iterator & start,
            int n,
            int * err,
            std::vector< unsigned char >::iterator end )
```

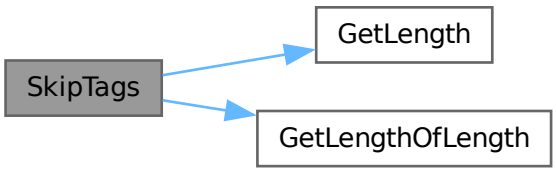
skips n BER attributes from char array, returns incremented iterator

Parameters

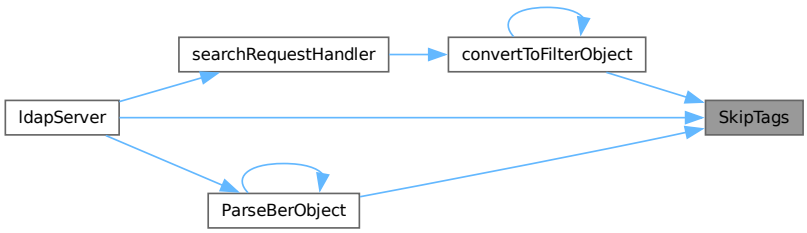
start	start of the tag
n	number of tags to skip
err	1 if error, 0 if success
end	end of the array

Definition at line 126 of file ber_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.11 ToLowerCase()

```
std::vector< unsigned char > ToLowerCase (
            std::vector< unsigned char > input )
```

converts std::vector<unsigned char> to lowercase

Parameters

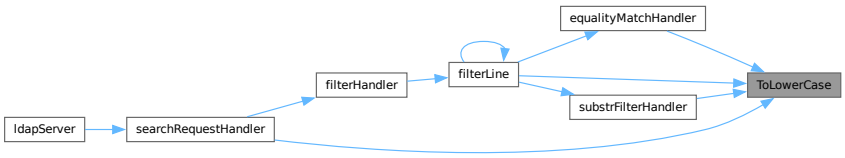
input	std::vector<unsigned char> to be converted

Returns

converted std::vector<unsigned char>

Definition at line 82 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.7.4.12 WriteIntAppend()

```
int WriteIntAppend (
            std::vector< unsigned char > & s,
            int value )
```

writes int in BER LDAP format to char array

Parameters

s	start of the string in char array
value	int to be written

Returns

-1 if error, 0 if success

Definition at line 169 of file ber_helper_functions.cpp.

Here is the caller graph for this function:

5.8 ber_helper_functions.h

Go to the documentation of this file.

```
00001
00007 #ifndef BER_HELPER_FUNCTIONS_H
00008 #define BER_HELPER_FUNCTIONS_H
00009 #include <vector>
00010 #include "inc/ber_constants.h"
00011 #include <stdio.h>
00012 #include <stdlib.h>
00013 #include <string>
00014
00020 std::vector<unsigned char> ToLowerCase(std::vector<unsigned char> input);
00021
00028 int ParseINT(unsigned char *s, int *err);
00029
00038 typedef enum filterTypes {
00039   AND,
00040   OR,
00041   NOT,
00042   equalityMatch,
00043   substrings,
00044   undefined,
00045 } filterTypes;
00046
00047 typedef enum berObjectTypes {
00048   berSequenceObject,
00049   berIntObject,
00050   berStringObject,
00051   berSetObject,
00052   berEnumObject,
00053   berBoolObject,
00054   berUndefined,
00055   berErr,
00056 } berObjectTypes;
00057
00064 int HowManyBytesWillIntUse(int value);
00065
00073 int WriteIntAppend(std::vector<unsigned char> &s, int value);
00074
00082 void AppendLenght4Bytes(std::vector<unsigned char> &start, int value);
00083
00092 int GetLength(std::vector<unsigned char>::iterator start, int *err,std::vector<unsigned
      char>::iterator end);
00093
00102 unsigned int ParseINT(std::vector<unsigned char>::iterator s, int *err,std::vector<unsigned
      char>::iterator end);
00103
00112 int GetLengthOfLength(std::vector<unsigned char>::iterator start, int *err,std::vector<unsigned
      char>::iterator end);
00113
00122 void SkipTags(std::vector<unsigned char>::iterator &start, int n, int *err, std::vector<unsigned
      char>::iterator end);
00123
00131 void GoIntoTag(std::vector<unsigned char>::iterator &start, int *err, std::vector<unsigned
      char>::iterator end);
00132
00141 void IncreaseLength4Bytes(std::vector<unsigned char>::iterator &start, int n,
00142                           int *err,std::vector<unsigned char>::iterator end);
00143
00150 filterTypes getFilterType(std::vector<unsigned char>::iterator start);
00151
00152 #endif
```

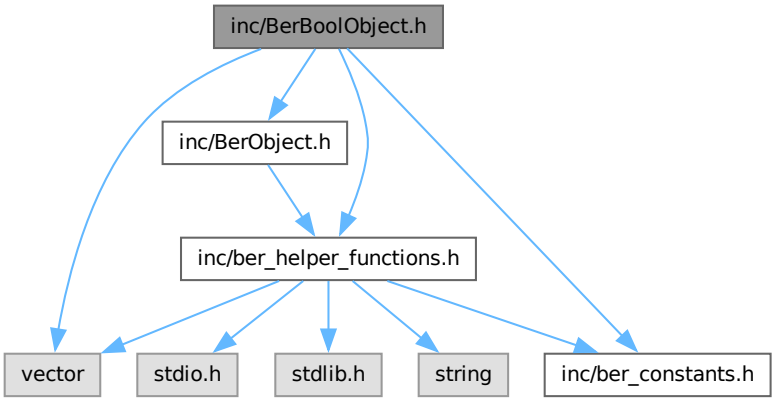
5.9 inc/BerBoolObject.h File Reference

Boolean object for BER LDAP.

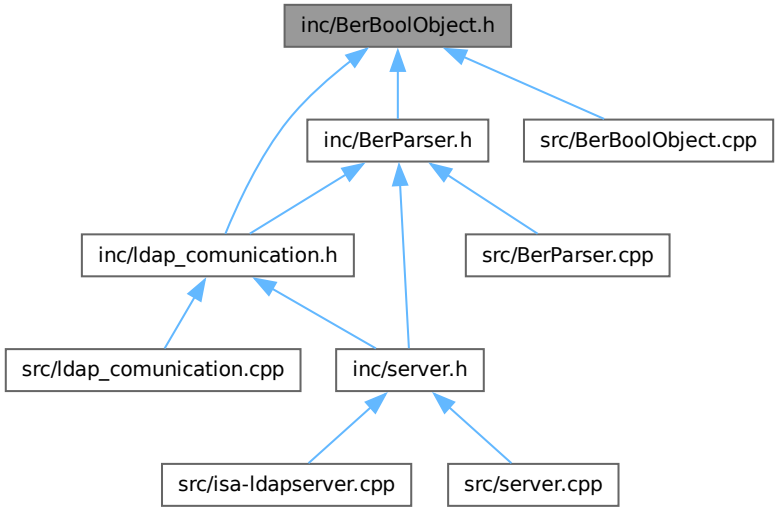
```
#include "inc/BerObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
```

```
#include <vector>
```

Include dependency graph for BerBoolObject.h:



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



Classes

- class BerBoolObject

5.9.1 Detailed Description

Boolean object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerBoolObject.h.

5.10 BerBoolObject.h

Go to the documentation of this file.

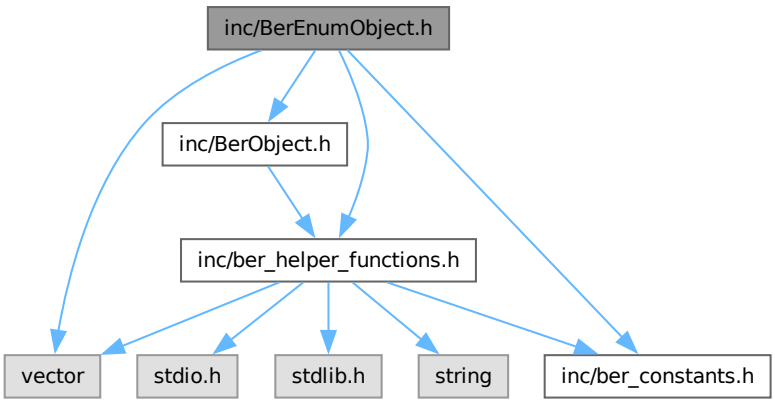
```
00001
00008 #ifndef BERBOOLOBJECT_H
00009 #define BERBOOLOBJECT_H
00010 #include "inc/BerObject.h"
00011 #include "inc/ber_constants.h"
00012 #include "inc/ber_helper_functions.h"
00013 #include <vector>
00014
00015 class BerBoolObject : public BerObject {
00016 private:
00017   bool value;
00018
00019 public:
00020   berObjectTypes getBerObjectType();
00021   long long int getLenght();
00022   std::vector<unsigned char> getBerRepresentation();
00023   BerBoolObject(char value);
00024   ~BerBoolObject();
00025 };
00026
00027 #endif
```

5.11 inc/BerEnumObject.h File Reference

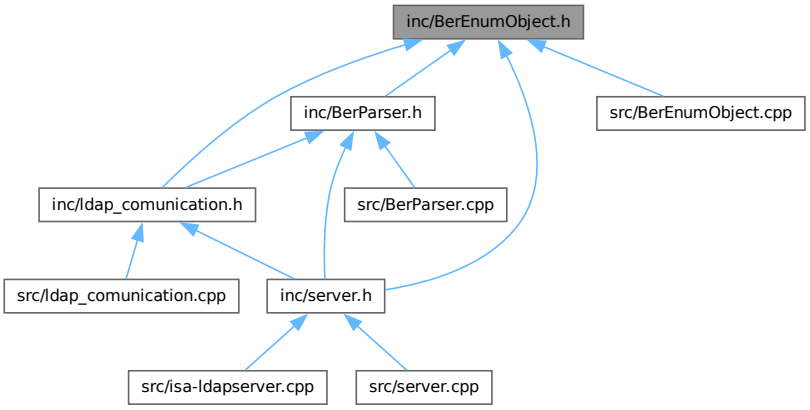
Enum object for BER LDAP.

```
#include "inc/BerObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for BerEnumObject.h:



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



Classes

- class BerEnumObject

5.11.1 Detailed Description

Enum object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerEnumObject.h.

5.12 BerEnumObject.h

Go to the documentation of this file.

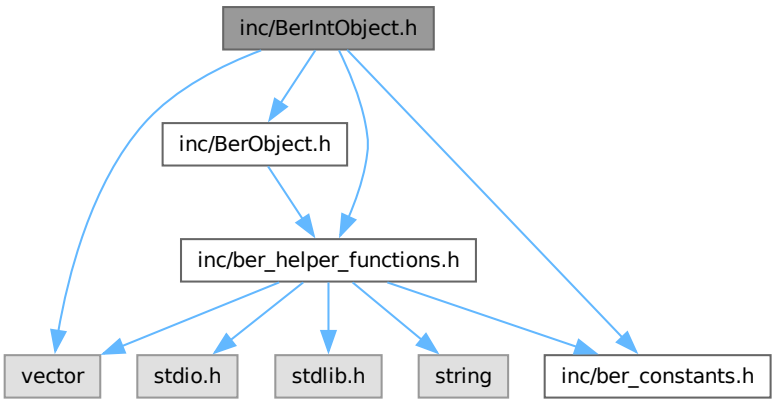
```
00001
00008 #ifndef BERENUMOBJECT_H
00009 #define BERENUMOBJECT_H
00010 #include "inc/BerObject.h"
00011 #include "inc/ber_constants.h"
00012 #include "inc/ber_helper_functions.h"
00013 #include <vector>
00014
00015 class BerEnumObject : public BerObject {
00016 private:
00017   int value;
00018
00019 public:
00020   berObjectTypes getBerObjectType();
00021   long long int getLenght();
00022   std::vector<unsigned char> getBerRepresentation();
00023   BerEnumObject(char value);
00024   ~BerEnumObject();
00025 };
00026
00027 #endif
```

5.13 inc/BerIntObject.h File Reference

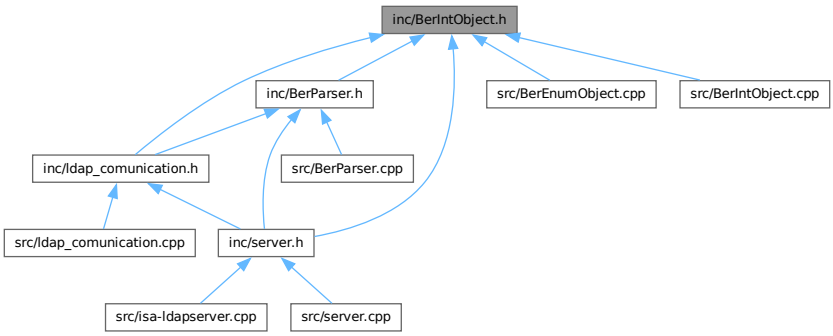
Integer object for BER LDAP.

```
#include "inc/BerObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for BerIntObject.h:



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



Classes

- class BerIntObject

5.13.1 Detailed Description

Integer object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerIntObject.h.

5.14 BerIntObject.h

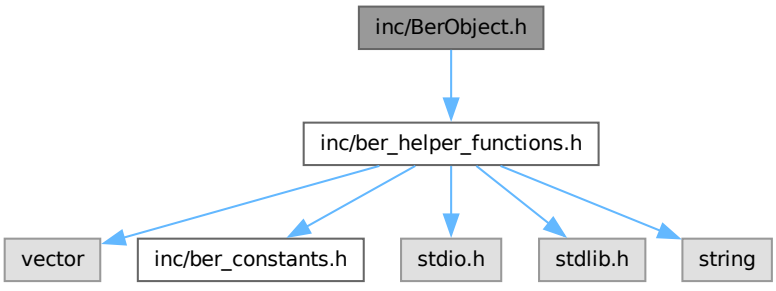
Go to the documentation of this file.

```
00001
00008 #ifndef BERINTOBJECT_H
00009 #define BERINTOBJECT_H
00010 #include "inc/BerObject.h"
00011 #include "inc/ber_constants.h"
00012 #include "inc/ber_helper_functions.h"
00013 #include <vector>
00014
00015 class BerIntObject : public BerObject {
00016 private:
00017   int value;
00018
00019 public:
00020   berObjectTypes getBerObjectType();
00021   int getValue();
00022   void setValue(int value);
00023   long long int getLenght();
00024   std::vector<unsigned char> getBerRepresentation();
00025   BerIntObject();
00026   BerIntObject(int value);
00027   ~BerIntObject();
00028 };
00029
00030 #endif
```

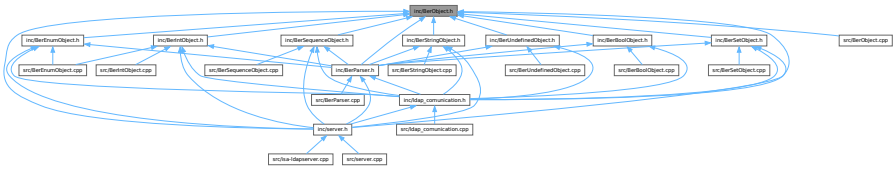
5.15 inc/BerObject.h File Reference

Base class for all BER objects.

`#include "inc/ber_helper_functions.h"`
Include dependency graph for BerObject.h:



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



Classes

- class BerObject

  Base class for all BER objects.

5.15.1 Detailed Description

Base class for all BER objects.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerObject.h.

5.16 BerObject.h

Go to the documentation of this file.

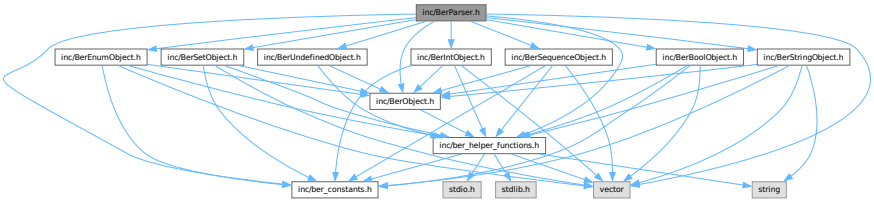
```
00001
00007 #ifndef BER_OBJECT_H
00008 #define BER_OBJECT_H
00009 #include "inc/ber_helper_functions.h"
00010
00015 class BerObject {
00016 public:
00022   virtual berObjectTypes getBerObjectType();
00029   virtual long long int getLenght();
00035   virtual std::vector<unsigned char> getBerRepresentation();
00036
00037   virtual ~BerObject();
00038 };
00039
00040 #endif
```

5.17 inc/BerParser.h File Reference

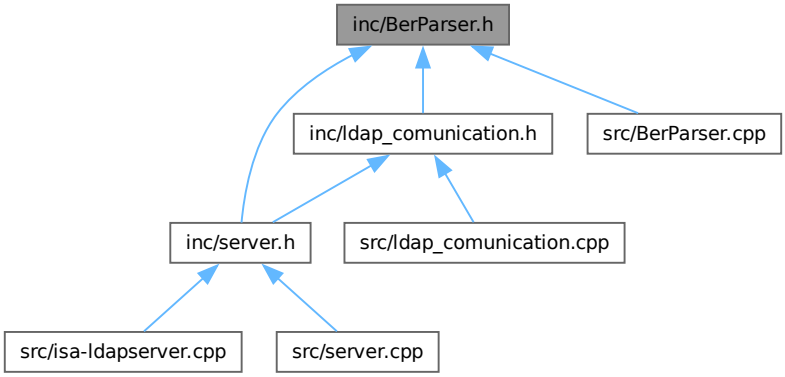
Parser for BER LDAP.

```
#include "inc/BerObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
#include "inc/BerEnumObject.h"
#include "inc/BerStringObject.h"
#include "inc/BerSetObject.h"
#include "inc/BerSequenceObject.h"
#include "inc/BerIntObject.h"
#include "inc/BerBoolObject.h"
#include "inc/BerUndefinedObject.h"
#include <vector>
```

Include dependency graph for BerParser.h:



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



Functions

- BerObject * ParseBerObject (std::vector< unsigned char >::iterator start, int *err, std::vector< unsigned char >::iterator end)

  Parses BER and converts it to BerObject.

5.17.1 Detailed Description

Parser for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerParser.h.

5.17.2 Function Documentation

5.17.2.1 ParseBerObject()

```
BerObject * ParseBerObject (
            std::vector< unsigned char >::iterator start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Parses BER and converts it to BerObject.

Parameters

start	start of the BER
err	1 if error, 0 if success
end	end of the array

Returns

BerObject*

Parses BER and converts it to BerObject.

Parameters

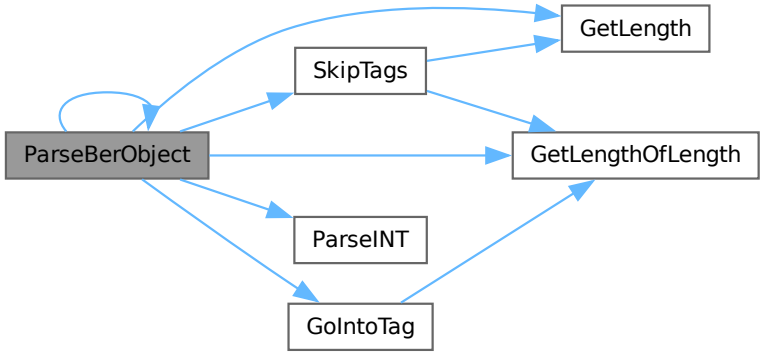
start	start of the BER
err	1 if error, 0 if success
end	end of the array

Returns

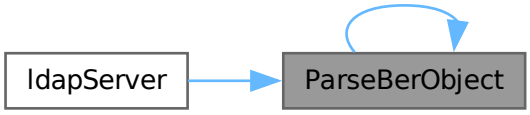
BerObject*

Definition at line 16 of file BerParser.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.18 BerParser.h

Go to the documentation of this file.

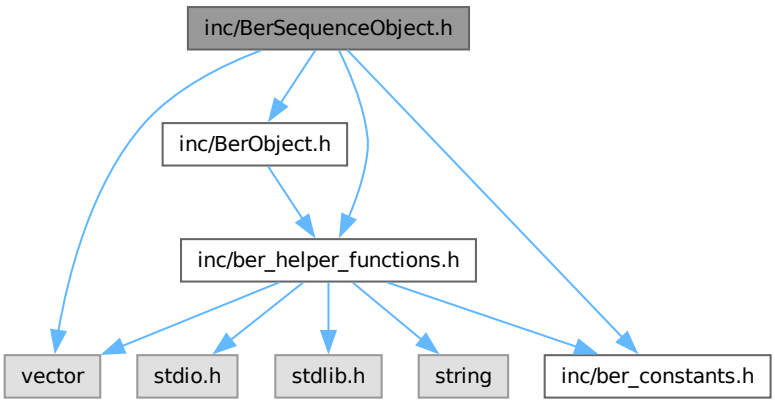
```
00001
00008 #ifndef BERPARSER_H
00009 #define BERPARSER_H
00010 #include "inc/BerObject.h"
00011 #include "inc/ber_constants.h"
00012 #include "inc/ber_helper_functions.h"
00013 #include "inc/BerEnumObject.h"
00014 #include "inc/BerStringObject.h"
00015 #include "inc/BerSetObject.h"
00016 #include "inc/BerSequenceObject.h"
00017 #include "inc/BerSetObject.h"
00018 #include "inc/BerIntObject.h"
00019 #include "inc/BerBoolObject.h"
00020 #include "inc/BerUndefinedObject.h"
00021 #include <vector>
00022
00031 BerObject *ParseBerObject(std::vector<unsigned char>::iterator start,
00032                           int *err,std::vector<unsigned char>::iterator end);
00033
00034 #endif
```

5.19 inc/BerSequenceObject.h File Reference

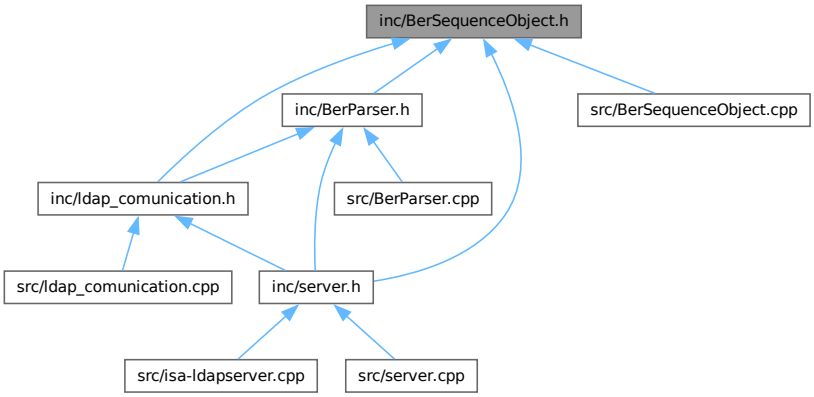
Sequence object for BER LDAP.

```
#include "inc/BerObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for BerSequenceObject.h:



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



Classes

- class BerSequenceObject

5.19.1 Detailed Description

Sequence object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerSequenceObject.h.

5.20 BerSequenceObject.h

Go to the documentation of this file.

```
00001
00008 #ifndef BERSEQUENCEOBJECT_H
00009 #define BERSEQUENCEOBJECT_H
00010 #include "inc/BerObject.h"
00011 #include "inc/ber_constants.h"
00012 #include "inc/ber_helper_functions.h"
00013
00014 #include <vector>
00015
00016 class BerSequenceObject : public BerObject {
00017 private:
00018   int tag;
00019
00020 public:
00021   std::vector<BerObject *> objects;
00022   berObjectTypes getBerObjectType();
00023   long long int getLenght();
00024   std::vector<unsigned char> getBerRepresentation();
00025   BerSequenceObject(int tag);
00026   BerSequenceObject();
00027   int GetTag();
00028   ~BerSequenceObject();
00029 };
00030
00031 #endif
```

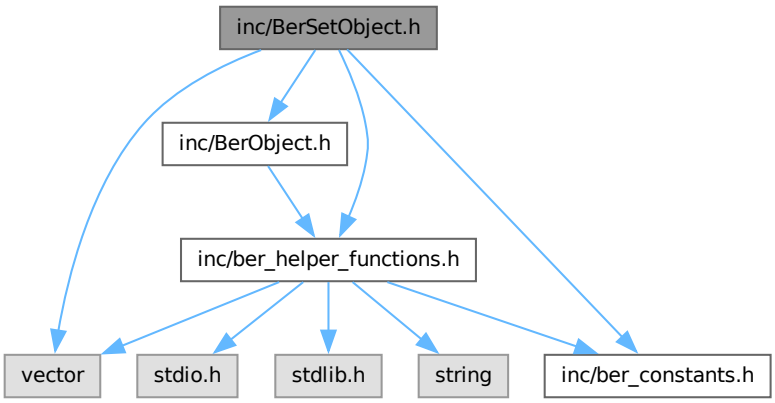
5.21 inc/BerSetObject.h File Reference

Set object for BER LDAP.

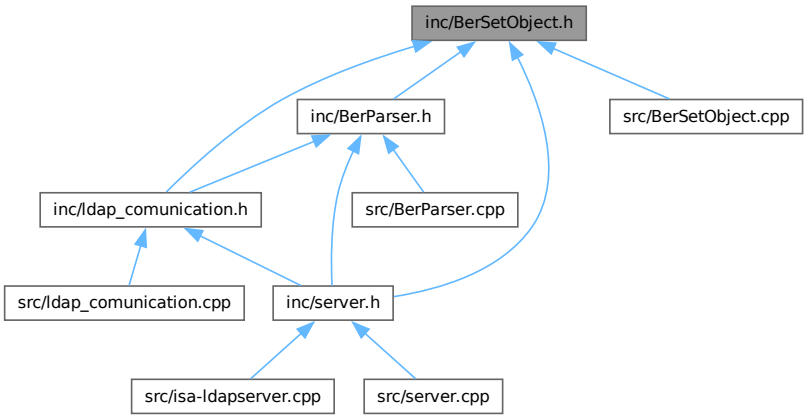
```
#include "inc/BerObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
```

```
#include <vector>
```

Include dependency graph for BerSetObject.h:



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



Classes

- class BerSetObject

5.21.1 Detailed Description

Set object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerSetObject.h.

5.22 BerSetObject.h

Go to the documentation of this file.

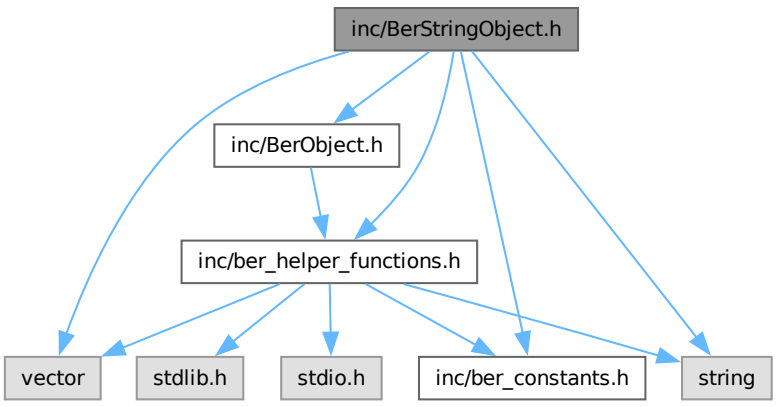
```
00001
00008 #ifndef BERSETOBJECT_H
00009 #define BERSETOBJECT_H
00010 #include "inc/BerObject.h"
00011 #include "inc/ber_constants.h"
00012 #include "inc/ber_helper_functions.h"
00013
00014 #include <vector>
00015
00016 class BerSetObject : public BerObject {
00017 public:
00018   std::vector<BerObject *> objects;
00019   berObjectTypes getBerObjectType();
00020   long long int getLenght();
00021   std::vector<unsigned char> getBerRepresentation();
00022   BerSetObject();
00023   ~BerSetObject();
00024 };
00025
00026 #endif
```

5.23 inc/BerStringObject.h File Reference

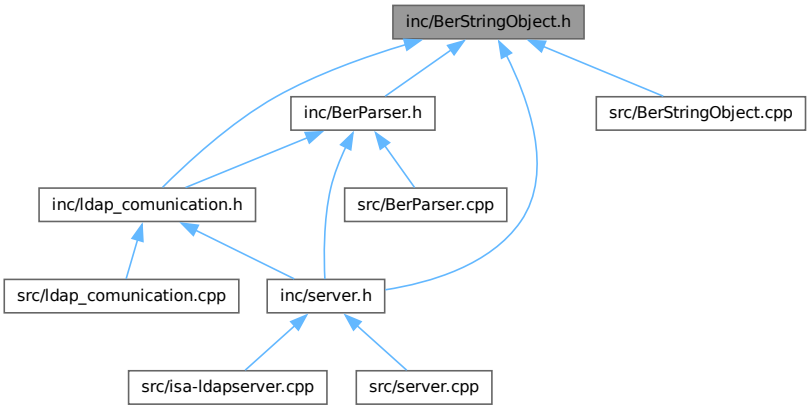
String object for BER LDAP.

```
#include "inc/BerObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
#include <string>
#include <vector>
```

Include dependency graph for BerStringObject.h:



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



Classes

- class BerStringObject

5.23.1 Detailed Description

String object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerStringObject.h.

5.24 BerStringObject.h

Go to the documentation of this file.

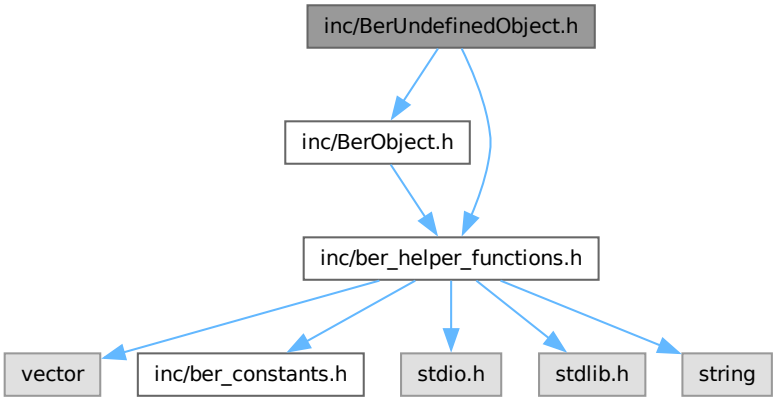
```
00001
00007 #ifndef BERSTRINGOBJECT_H
00008 #define BERSTRINGOBJECT_H
00009 #include "inc/BerObject.h"
00010 #include "inc/ber_constants.h"
00011 #include "inc/ber_helper_functions.h"
00012 #include <string>
00013 #include <vector>
00014
00015 class BerStringObject : public BerObject {
00016 public:
00017   berObjectTypes getBerObjectType();
00018   std::vector<unsigned char> value;
00019   long long int getLenght();
00020   std::vector<unsigned char> getBerRepresentation();
00021   BerStringObject();
00022   BerStringObject(std::vector<unsigned char> value);
00023 BerStringObject(std::string value);
00024 };
00025
00026 #endif
```

5.25 inc/BerUndefinedObject.h File Reference

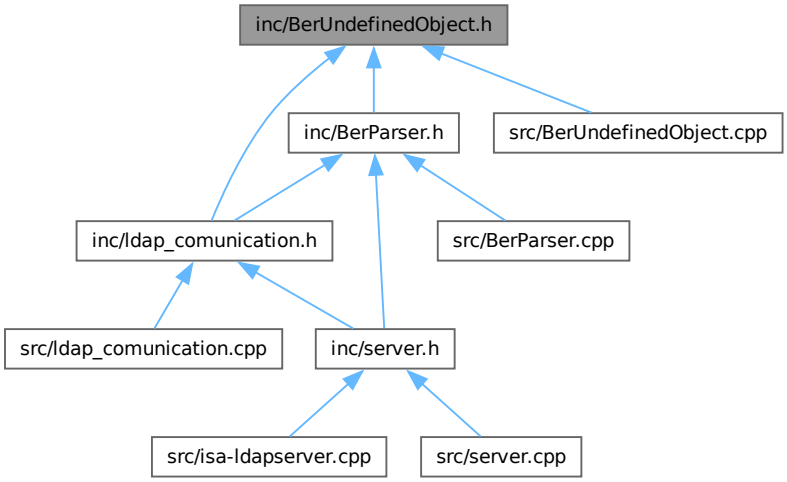
Undefined object for BER LDAP, for containing unknown data.

```
#include "inc/BerObject.h"
#include "inc/ber_helper_functions.h"
```

Include dependency graph for BerUndefinedObject.h:



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



Classes

- class BerUndefinedObject

5.25.1 Detailed Description

Undefined object for BER LDAP, for containing unknown data.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerUndefinedObject.h.

5.26 BerUndefinedObject.h

Go to the documentation of this file.

```
00001
00007 #ifndef BERUNDEFINEDOBJECT_H
00008 #define BERUNDEFINEDOBJECT_H
00009 #include "inc/BerObject.h"
00010 #include "inc/ber_helper_functions.h"
00011
00012 class BerUndefinedObject : public BerObject {
00013 private:
00014   std::vector<unsigned char> value;
00015
00016 public:
00017   berObjectTypes getBerObjectType();
00018   long long int getLenght();
00019   std::vector<unsigned char> getBerRepresentation();
00020   BerUndefinedObject(std::vector<unsigned char> value);
00021 };
00022
00023 #endif
```

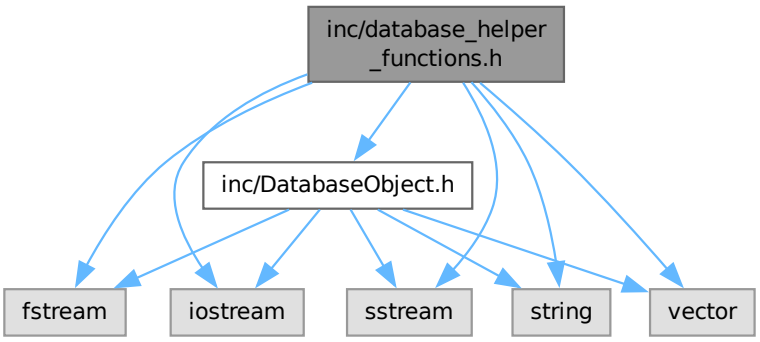
5.27 inc/database_helper_functions.h File Reference

Helper functions for database.

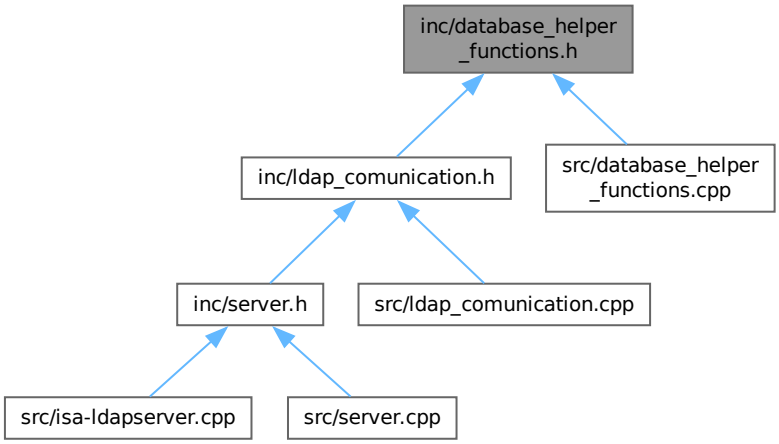
```
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
```

```
#include "inc/DatabaseObject.h"
```

Include dependency graph for database_helper_functions.h:



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



Functions

- std::vector< DatabaseObject > removeDuplicates (std::vector< DatabaseObject > input)

  Removes dupplicates from vector of DatabaseObjects.

5.27.1 Detailed Description

Helper functions for database.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file database_helper_functions.h.

5.27.2 Function Documentation

5.27.2.1 removeDuplicates()

```
std::vector< DatabaseObject > removeDuplicates (
            std::vector< DatabaseObject > input )
```

Removes dupplicates from vector of DatabaseObjects.

Parameters

input	

Returns

std::vector<DatabaseObject>

Definition at line 9 of file database_helper_functions.cpp.

Here is the caller graph for this function:

5.28 database_helper_functions.h

Go to the documentation of this file.

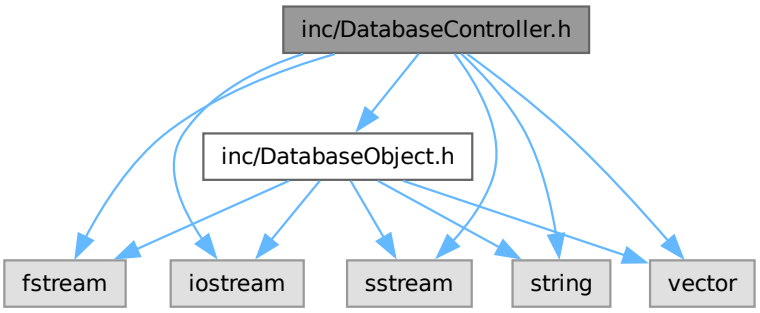
```
00001
00007 #ifndef DATABASE_HELPER_FUNCTIONS_H
00008 #define DATABASE_HELPER_FUNCTIONS_H
00009 #include <fstream>
00010 #include <iostream>
00011 #include <sstream>
00012 #include <string>
00013 #include <vector>
00014 #include "inc/DatabaseObject.h"
00015
00022 std::vector<DatabaseObject> removeDuplicates(std::vector<DatabaseObject> input);
00023
00024 #endif
```

5.29 inc/DatabaseController.h File Reference

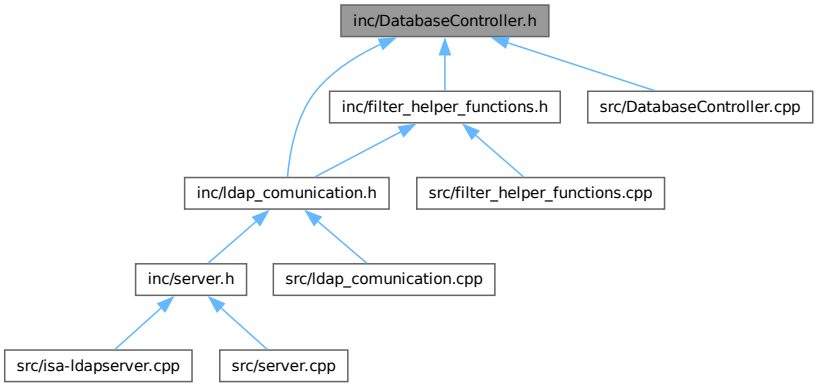
Controller for database csv file.

```
#include "inc/DatabaseObject.h"
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
```

Include dependency graph for DatabaseController.h:



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



Classes

- class DatabaseController

  class for loading and parsing database file

5.29.1 Detailed Description

Controller for database csv file.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file DatabaseController.h.

5.30 DatabaseController.h

Go to the documentation of this file.

```
00001
00007 #ifndef DATABASECONTROLLER_H
00008 #define DATABASECONTROLLER_H
00009 #include "inc/DatabaseObject.h"
00010 #include <fstream>
00011 #include <iostream>
00012 #include <sstream>
00013 #include <string>
00014 #include <vector>
00015
00020 class DatabaseController {
00021 private:
00022   std::ifstream file;
00023   std::vector<unsigned char> sanitaze(std::vector<unsigned char> input);
00024
00025 public:
00032   DatabaseObject loadNextRow(int *err);
00033
00039   std::vector<DatabaseObject> loadAllRows();
00040
00046   DatabaseController(std::string fileName);
00047   ~DatabaseController();
00048 };
00049
00050 #endif
```

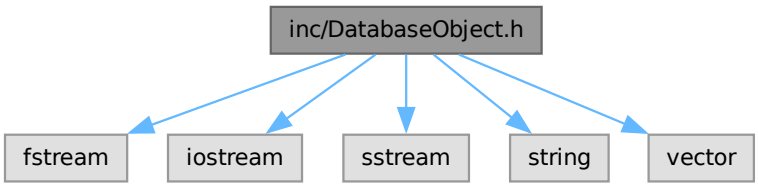
5.31 inc/DatabaseObject.h File Reference

Object representing one row from database.

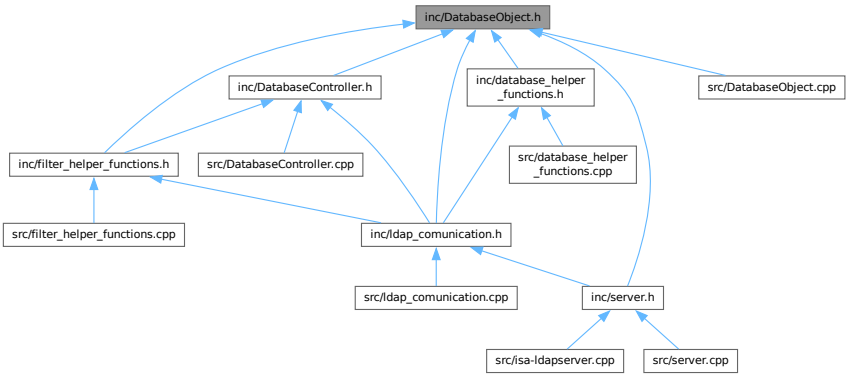
```
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
```

```
#include <vector>
```

Include dependency graph for DatabaseObject.h:



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



Classes

- class DatabaseObject

  Object representing one row from database.

5.31.1 Detailed Description

Object representing one row from database.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file DatabaseObject.h.

5.32 DatabaseObject.h

Go to the documentation of this file.

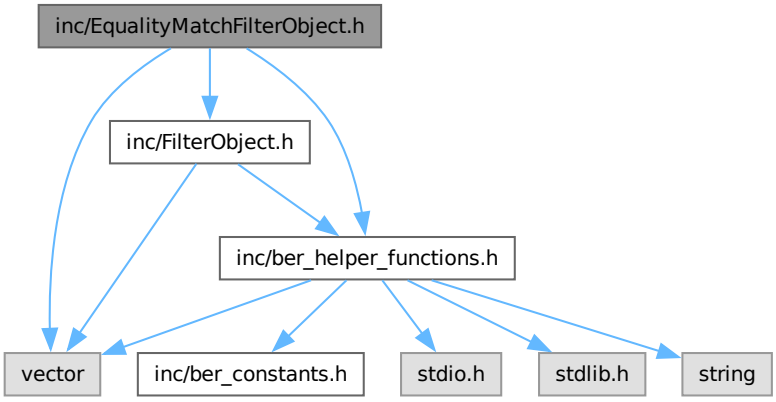
```
00001
00007 #ifndef DATABASE_OBJECT_H
00008 #define DATABASE_OBJECT_H
00009 #include <fstream>
00010 #include <iostream>
00011 #include <sstream>
00012 #include <string>
00013 #include <vector>
00014
00019 class DatabaseObject {
00020 private:
00021   std::vector<unsigned char> name;
00022   std::vector<unsigned char> uid;
00023   std::vector<unsigned char> email;
00024
00025 public:
00026   std::vector<unsigned char> get_name();
00027   std::vector<unsigned char> get_uid();
00028   std::vector<unsigned char> get_email();
00029   DatabaseObject(std::vector<unsigned char> name,
00030                  std::vector<unsigned char> uid,
00031                  std::vector<unsigned char> email);
00032 };
00033
00034
00035 #endif
```

5.33 inc/EqualityMatchFilterObject.h File Reference

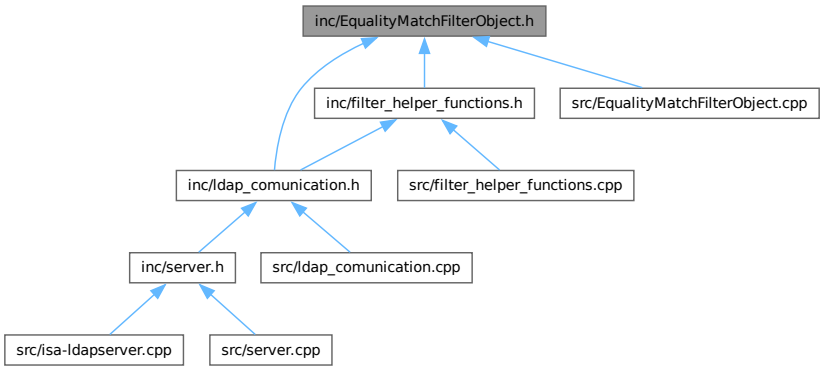
Equality match filter object for BER LDAP.

```
#include "inc/FilterObject.h"
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for EqualityMatchFilterObject.h:



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



Classes

- class EqualityMatchFilter

5.33.1 Detailed Description

Equality match filter object for BER LDAP.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file EqualityMatchFilterObject.h.

5.34 EqualityMatchFilterObject.h

Go to the documentation of this file.

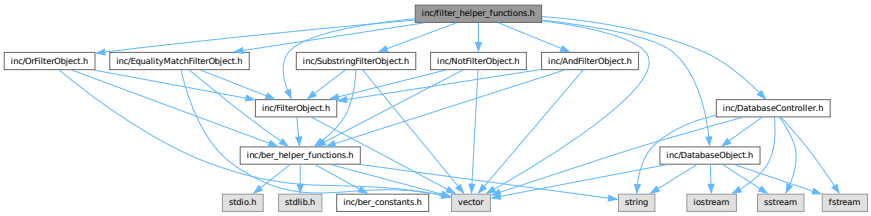
```
00001
00008 #ifndef EQUALITYMATCHFILTEROBJECT_H
00009 #define EQUALITYMATCHFILTEROBJECT_H
00010 #include "inc/FilterObject.h"
00011 #include "inc/ber_helper_functions.h"
00012
00013 #include <vector>
00014
00015 class EqualityMatchFilter : public FilterObject {
00016 private:
00017   std::vector<unsigned char> attributeDescription;
00018   std::vector<unsigned char> assertionValue;
00019
00020 public:
00021   EqualityMatchFilter(std::vector<unsigned char> attributeDescription,
00022                       std::vector<unsigned char> assertionValue);
00023   std::vector<unsigned char> getAttributeDescription();
00024   std::vector<unsigned char> getAssertionValue();
00025   filterTypes getFilterType();
00026 };
00027 #endif
```

5.35 inc/filter_helper_functions.h File Reference

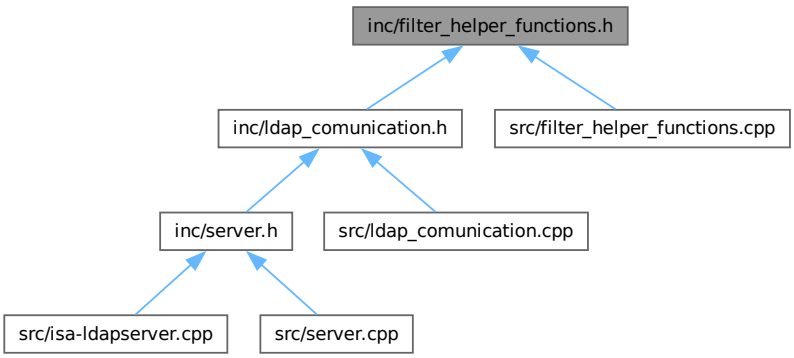
Helper functions for filters.

```
#include "inc/FilterObject.h"
#include "inc/NotFilterObject.h"
#include "inc/AndFilterObject.h"
#include "inc/OrFilterObject.h"
#include "inc/EqualityMatchFilterObject.h"
#include "inc/SubstringFilterObject.h"
#include "inc/DatabaseObject.h"
#include "inc/DatabaseController.h"
#include "vector"
```

Include dependency graph for filter_helper_functions.h:



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



Functions

- bool substrFilterHandler (SubstringFilter *sf, int *err, std::vector< unsigned char > attribute)

  evaluates if filter is true for given database entry
- bool equalityMatchHandler (EqualityMatchFilter *emf, int *err, std::vector< unsigned char > attribute)

  evalues if filter is true for given database entry
- bool filterLine (FilterObject *f, int *err, DatabaseObject &databaseEntry)

  evaluates if filter is true for given database entry
- std::vector< DatabaseObject > filterHandler (FilterObject *f, int *err, const char *dbLocation, int sizeLimit)

  evaluates if filter is true for given database entries
- FilterObject * convertToFilterObject (std::vector< unsigned char >::iterator BERfilter, std::vector< unsigned char >::iterator end)

  converts BER representation of filters to filter object

5.35.1 Detailed Description

Helper functions for filters.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file filter_helper_functions.h.

5.35.2 Function Documentation

5.35.2.1 convertToFilterObject()

```
FilterObject * convertToFilterObject (
            std::vector< unsigned char >::iterator BERfilter,
            std::vector< unsigned char >::iterator end )
```

converts BER representation of filters to filter object

Parameters

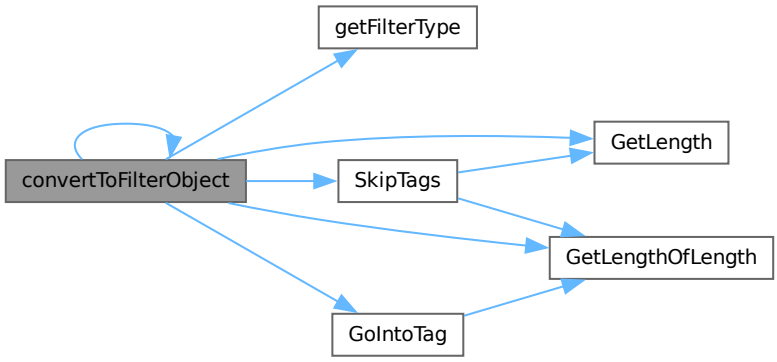
BERfilter	start of the BER filter
end	end of the BER filter

Returns

FilterObject*

Definition at line 204 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.35.2.2 equalityMatchHandler()

```
bool equalityMatchHandler (
            EqualityMatchFilter * emf,
            int * err,
            std::vector< unsigned char > attribute )
```

evalues if filter is true for given database entry

Parameters

emf	equality match filter object
err	1 if error, 0 if success
attribute	attribute to be filtered

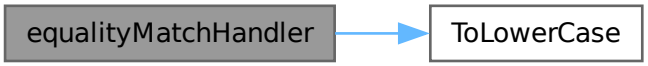
Returns

true

false

Definition at line 83 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.35.2.3 filterHandler()

```
std::vector< DatabaseObject > filterHandler (
            FilterObject * f,
            int * err,
            const char * dbLocation,
            int sizeLimit )
```

evaluates if filter is true for given database entries

Parameters

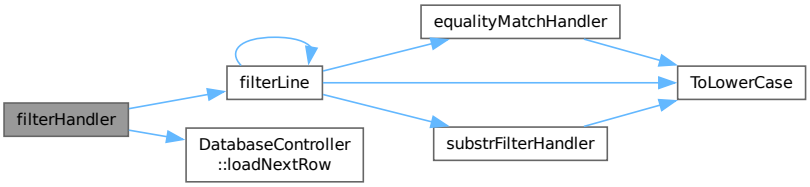
f	filter object
err	1 if error, 0 if success
dbLocation	path to database file
sizeLimit	maximum number of entries to be returned

Returns

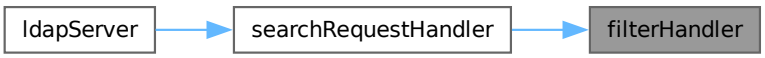
std::vector<DatabaseObject>

Definition at line 175 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.35.2.4 filterLine()

```
bool filterLine (
            FilterObject * f,
            int * err,
            DatabaseObject & databaseEntry )
```

evaluates if filter is true for given database entry

Parameters

f	filter object
err	1 if error, 0 if success
databaseEntry	database entry to be filtered

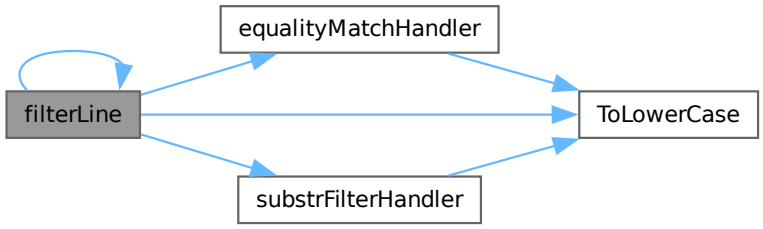
Returns

true

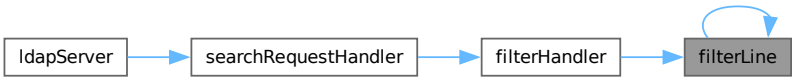
false

Definition at line 91 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.35.2.5 substrFilterHandler()

```
bool substrFilterHandler (
            SubstringFilter * sf,
            int * err,
            std::vector< unsigned char > attribute )
```

evaluates if filter is true for given database entry

Parameters

sf	substring filter object
err	1 if error, 0 if success
attribute	attribute to be filtered

Returns

true

false

Definition at line 8 of file filter_helper_functions.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.36 filter_helper_functions.h

Go to the documentation of this file.

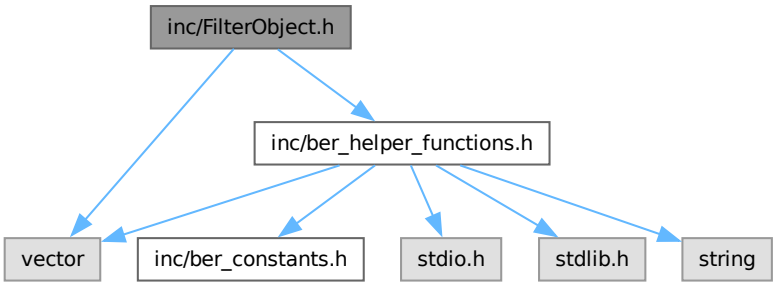
```
00001
00007 #ifndef INC_FILTER_HELPER_FUNCTIONS_H
00008 #define INC_FILTER_HELPER_FUNCTIONS_H
00009
00010
00011 #include "inc/FilterObject.h"
00012 #include "inc/NotFilterObject.h"
00013 #include "inc/AndFilterObject.h"
00014 #include "inc/OrFilterObject.h"
00015 #include "inc/EqualityMatchFilterObject.h"
00016 #include "inc/SubstringFilterObject.h"
00017 #include "inc/DatabaseObject.h"
00018 #include "inc/DatabaseController.h"
00019 #include "vector"
00020
00030 bool substrFilterHandler(SubstringFilter *sf, int *err,
00031                          std::vector<unsigned char> attribute);
00032
00042 bool equalityMatchHandler(EqualityMatchFilter *emf, int *err,
00043                           std::vector<unsigned char> attribute) ;
00044
00054 bool filterLine(FilterObject *f, int *err, DatabaseObject &databaseEntry) ;
00055
00065 std::vector<DatabaseObject>
00066 filterHandler(FilterObject *f, int *err, const char *dbLocation, int sizeLimit);
00067
00075 FilterObject *convertToFilterObject(std::vector<unsigned char>::iterator BERfilter,
      std::vector<unsigned char>::iterator end);
00076
00077 #endif
```

5.37 inc/FilterObject.h File Reference

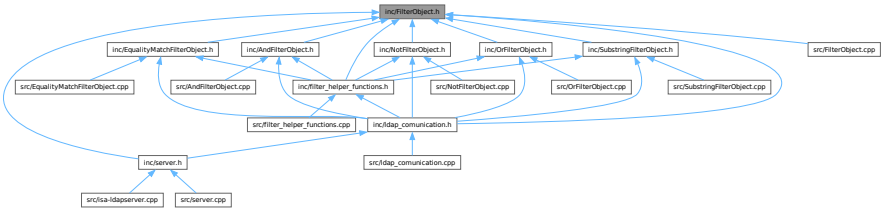
Base class for all filter objects.

```
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for FilterObject.h:



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



Classes

- class FilterObject

  base class for all filter objects

5.37.1 Detailed Description

Base class for all filter objects.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file FilterObject.h.

5.38 FilterObject.h

Go to the documentation of this file.

```
00001
00007 #ifndef FILTER_OBJECT_H
00008 #define FILTER_OBJECT_H
00009 #include "inc/ber_helper_functions.h"
00010
00011 #include <vector>
00012
00017 class FilterObject {
00018 public:
00019   virtual filterTypes getFilterType();
00020   virtual ~FilterObject();
00021 };
00022
00023
00024 #endif
```

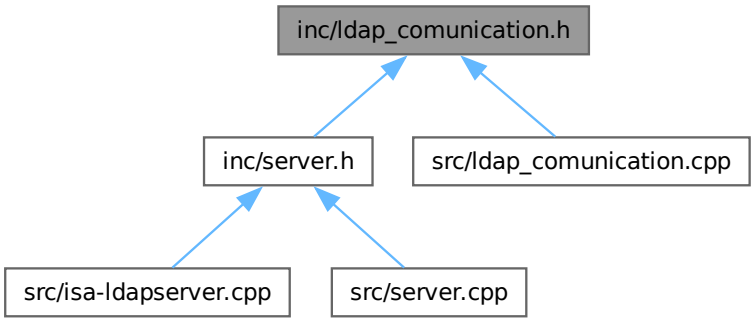
5.39 inc/ldap_comunication.h File Reference

Functions for communication with ldap client.

```
#include "inc/AndFilterObject.h"
#include "inc/BerBoolObject.h"
#include "inc/BerEnumObject.h"
#include "inc/BerIntObject.h"
#include "inc/BerObject.h"
#include "inc/BerParser.h"
#include "inc/BerSequenceObject.h"
#include "inc/BerSetObject.h"
#include "inc/BerStringObject.h"
#include "inc/BerUndefinedObject.h"
#include "inc/DatabaseController.h"
#include "inc/DatabaseObject.h"
#include "inc/EqualityMatchFilterObject.h"
#include "inc/FilterObject.h"
#include "inc/NotFilterObject.h"
#include "inc/OrFilterObject.h"
#include "inc/SubstringFilterObject.h"
#include "inc/ber_constants.h"
#include "inc/ber_helper_functions.h"
#include "inc/database_helper_functions.h"
#include "inc/filter_helper_functions.h"
#include <algorithm>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <string>
#include <sys/resource.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
```

`#include <vector>`
Include dependency graph for ldap_comunication.h:

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



Classes

- struct searchedAttributes
- struct searchRequest

Typedefs

- typedef struct searchedAttributes **searchedAttributesType**
- typedef struct searchRequest **searchRequestType**

Enumerations

- enum **atributeDescriptions** { **cn** , **email** , **uid** }

Functions

- BerObject * InitSearchResultEntry (BerObject *searchRequest, std::vector< unsigned char > LDAPDN)

  Initialize the search result entry envelope.
- int AddToSearchResultEntry (BerObject *envelope, std::vector< unsigned char > &attributeDescription, std::vector< unsigned char > &attributeValue)

  Adds an attribute to the search result entry envelope.
- int checkSearchRequest (BerObject *searchRequest)

  checks if the search request is valid
- int sendNoticeOfDisconnection (int comSocket, char errCode)

sends notice of disconnection to the client

- int searchRequestHandler (BerObject *searchRequest, int comm_socket, const char *dbPath)

  sends search result entry to the client

- BerObject * CreateBindResponse (BerObject *bindRequest, int resultCode)

  Create a Bind Response object.

- int loadEnvelope (std::vector< unsigned char > &bindRequest, int comm_socket)

  loads the envelope from the client, waits until all the date are received

- int sendSearchResultDone (BerSequenceObject *searchRequest, int comm_socket, unsigned int result_↩ code)

  sends the search result done envelope to the client

5.39.1 Detailed Description

Functions for communication with ldap client.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file ldap_comunication.h.

5.39.2 Enumeration Type Documentation

5.39.2.1 atributeDescriptions

```
enum atributeDescriptions
```

Definition at line 52 of file ldap_comunication.h.

5.39.3 Function Documentation

5.39.3.1 AddToSearchResultEntry()

```
int AddToSearchResultEntry (
            BerObject * envelope,
            std::vector< unsigned char > & attributeDescription,
            std::vector< unsigned char > & attributeValue )
```

Adds an attribute to the search result entry envelope.

Parameters

envelope	search result entry envelope
attributeDescription	
attributeValue	

Returns

int

Definition at line 24 of file ldap_comunication.cpp.

Here is the caller graph for this function:

5.39.3.2 checkSearchRequest()

```
int checkSearchRequest (
            BerObject * searchRequest )
```

checks if the search request is valid

Parameters

searchRequest	

Returns

int 0 if valid, -1 if inavalid application sequence, -2 if invalid message id or whole envelope

Definition at line 74 of file ldap_comunication.cpp.

Here is the caller graph for this function:

5.39.3.3 CreateBindResponse()

```
BerObject * CreateBindResponse (
            BerObject * bindRequest,
            int resultCode )
```

Create a Bind Response object.

Parameters

bindRequest	
resultCode	

Returns

BerObject*

Definition at line 43 of file ldap_comunication.cpp.

Here is the caller graph for this function:

5.39.3.4 InitSearchResultEntry()

```
BerObject * InitSearchResultEntry (
            BerObject * searchRequest,
            std::vector< unsigned char > LDAPDN )
```

Initialize the search result entry envelope.

Parameters

searchRequest	search request envelope for which the search result entry
LDAPDN	LDAPDN of the entry

Returns

BerObject*

Definition at line 9 of file ldap_comunication.cpp.

Here is the caller graph for this function:

5.39.3.5 loadEnvelope()

```
int loadEnvelope (
            std::vector< unsigned char > & bindRequest,
            int comm_socket )
```

loads the envelope from the client, waits until all the date are received

Parameters

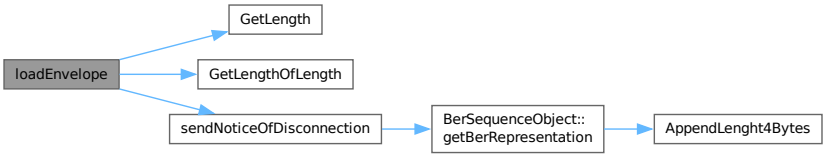
bindRequest	returns the envelope as a vector of unsigned chars
comm_socket	socket to receive the envelope from

Returns

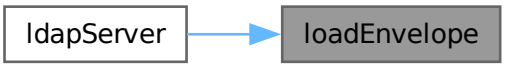
int 0 if success, -1 if error ocured

Definition at line 275 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.6 searchRequestHandler()

```
int searchRequestHandler (
            BerObject * searchRequest,
            int comm_socket,
            const char * dbPath )
```

sends search result entry to the client

Parameters

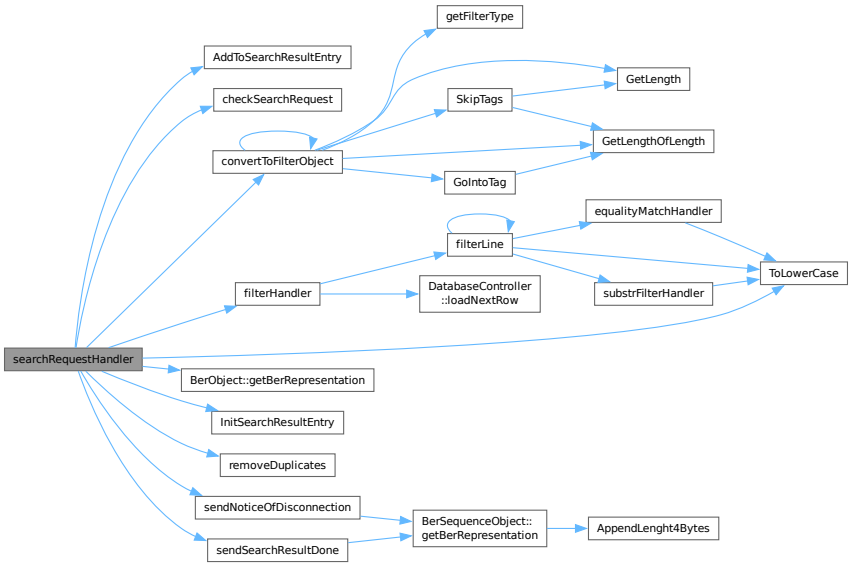
envelope	search request envelope
comSocket	socket to send the envelope to

Returns

int

Definition at line 140 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.39.3.7 sendNoticeOfDisconnection()

```
int sendNoticeOfDisconnection (
            int comSocket,
            char errCode )
```

sends notice of disconnection to the client

Parameters

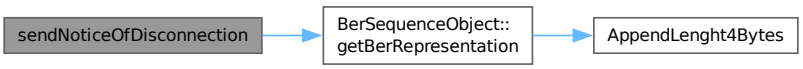
comSocket	socket to send the notice to
errCode	error code

Returns

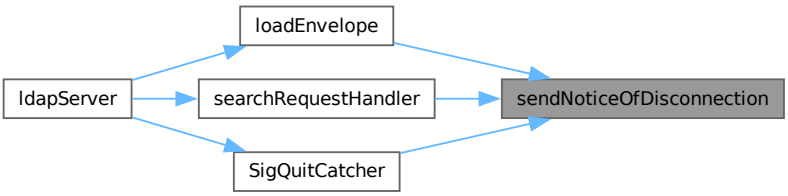
int

Definition at line 125 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.8 sendSearchResultDone()

```
int sendSearchResultDone (
            BerSequenceObject * searchRequest,
            int comm_socket,
            unsigned int result_code )
```

sends the search result done envelope to the client

Parameters

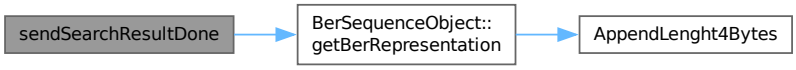
searchRequest	search request envelope for which the search result done is
comm_socket	socket to send the envelope to
result_code	

Returns

int

Definition at line 55 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.40 ldap_comunication.h

Go to the documentation of this file.

```
00001
00007 #ifndef LDAP_COMUNICATION_H
00008 #define LDAP_COMUNICATION_H
00009 #include "inc/AndFilterObject.h"
00010 #include "inc/BerBoolObject.h"
00011 #include "inc/BerEnumObject.h"
00012 #include "inc/BerIntObject.h"
00013 #include "inc/BerObject.h"
00014 #include "inc/BerParser.h"
00015 #include "inc/BerSequenceObject.h"
00016 #include "inc/BerSetObject.h"
00017 #include "inc/BerStringObject.h"
00018 #include "inc/BerUndefinedObject.h"
00019 #include "inc/DatabaseController.h"
00020 #include "inc/DatabaseObject.h"
00021 #include "inc/EqualityMatchFilterObject.h"
00022 #include "inc/FilterObject.h"
00023 #include "inc/NotFilterObject.h"
00024 #include "inc/OrFilterObject.h"
00025 #include "inc/SubstringFilterObject.h"
00026 #include "inc/ber_constants.h"
00027 #include "inc/ber_helper_functions.h"
00028 #include "inc/database_helper_functions.h"
00029 #include "inc/filter_helper_functions.h"
00030 #include <algorithm>
00031 #include <arpa/inet.h>
00032 #include <netinet/in.h>
00033 #include <stdio.h>
00034 #include <stdlib.h>
00035 #include <string.h>
00036 #include <string>
00037 #include <sys/resource.h>
00038 #include <sys/socket.h>
00039 #include <sys/time.h>
00040 #include <sys/types.h>
00041 #include <sys/wait.h>
00042 #include <unistd.h>
00043 #include <vector>
```

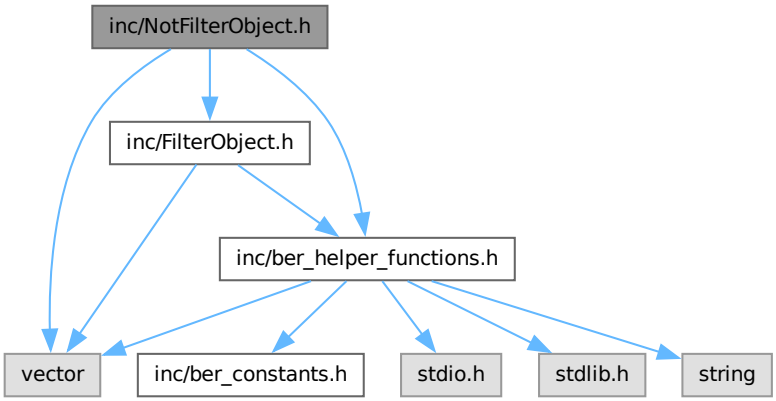
```
00044
00045 typedef struct searchedAttributes {
00046   bool cn;
00047   bool email;
00048   bool uid;
00049 } searchedAttributesType;
00050
00051 // enum for attributes (cn, email, uid)
00052 typedef enum { cn, email, uid } atributeDescriptions;
00053
00054 // sequence - envelope
00055 //     int    - message ID
00056 //     application 3 - search request
00057 //     octed string - base object
00058 //     enum - scope
00059 //     enum - derefAliases
00060 //     int - sizeLimit
00061 //     int - timeLimit
00062 //     bool - typesOnly
00063 //     sequence - FilterObject
00064 //     sequence - attributes
00065
00066 typedef struct searchRequest {
00067   int messageIDLength;
00068   unsigned int sizeLimit;
00069   searchedAttributesType attributes;
00070 } searchRequestType;
00071
00079 BerObject *InitSearchResultEntry(BerObject *searchRequest,
00080                                  std::vector<unsigned char> LDAPDN);
00081
00090 int AddToSearchResultEntry(BerObject *envelope,
00091                            std::vector<unsigned char> &attributeDescription,
00092                            std::vector<unsigned char> &attributeValue);
00099 int checkSearchRequest(BerObject *searchRequest);
00100
00108 int sendNoticeOfDisconnection(int comSocket, char errCode);
00109
00117 int searchRequestHandler(BerObject *searchRequest, int comm_socket,
00118                          const char *dbPath);
00119
00127 BerObject *CreateBindResponse(BerObject *bindRequest, int resultCode);
00128
00136 int loadEnvelope(std::vector<unsigned char> &bindRequest, int comm_socket) ;
00137
00146 int sendSearchResultDone(BerSequenceObject *searchRequest, int comm_socket,
00147                          unsigned int result_code);
00148 #endif
```

5.41 inc/NotFilterObject.h File Reference

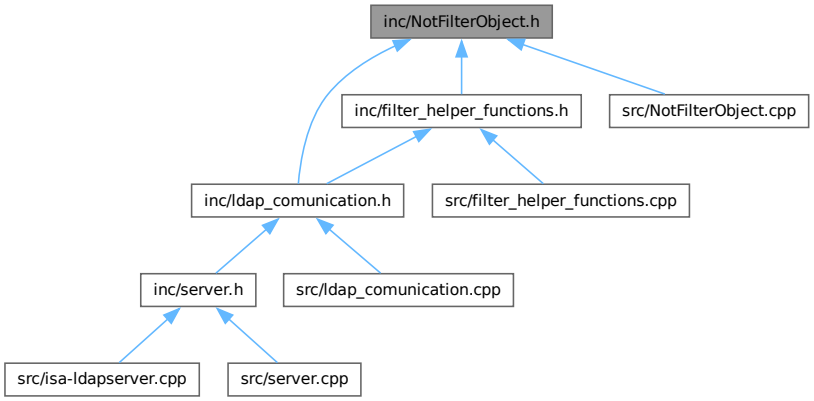
Helper functions for parsing arguments.

```
#include "inc/FilterObject.h"
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for NotFilterObject.h:



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



Classes

- class NotFilter

5.41.1 Detailed Description

Helper functions for parsing arguments.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file NotFilterObject.h.

5.42 NotFilterObject.h

Go to the documentation of this file.

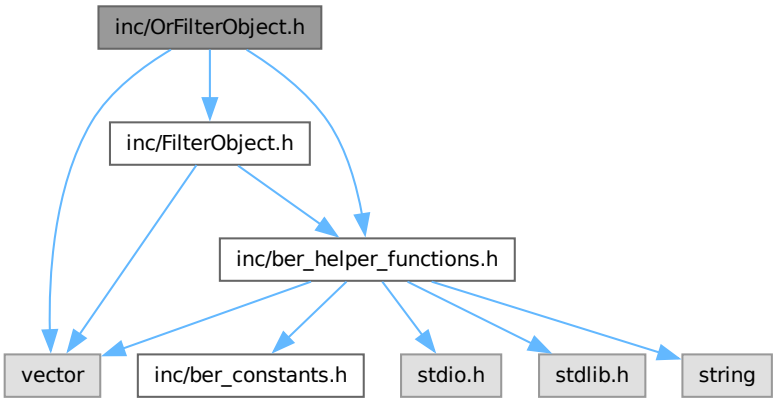
```
00001
00007 #ifndef NOTFILTEROBJECT_H
00008 #define NOTFILTEROBJECT_H
00009 #include "inc/FilterObject.h"
00010 #include "inc/ber_helper_functions.h"
00011
00012 #include <vector>
00013
00014 class NotFilter : public FilterObject {
00015 public:
00016   FilterObject *filter;
00017   filterTypes getFilterType();
00018   ~NotFilter();
00019 };
00020
00021 #endif
```

5.43 inc/OrFilterObject.h File Reference

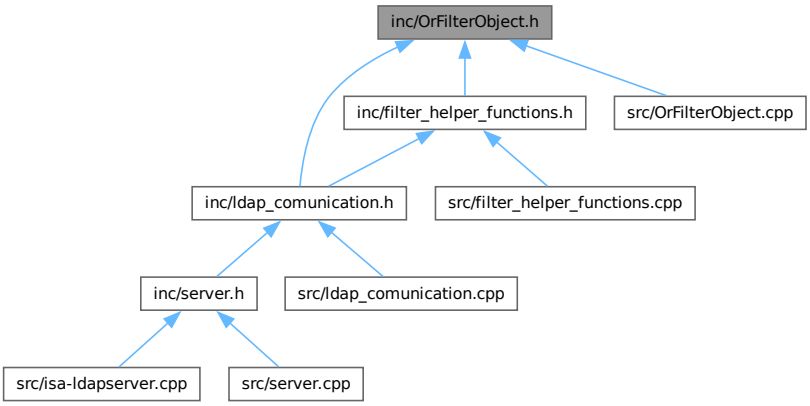
Object for OR filter.

```
#include "inc/FilterObject.h"
#include "inc/ber_helper_functions.h"
#include <vector>
```

Include dependency graph for OrFilterObject.h:



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



Classes

- class OrFilter

5.43.1 Detailed Description

Object for OR filter.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file OrFilterObject.h.

5.44 OrFilterObject.h

Go to the documentation of this file.

```
00001
00007 #ifndef ORFILTEROBJECT_H
00008 #define ORFILTEROBJECT_H
00009 #include "inc/FilterObject.h"
00010 #include "inc/ber_helper_functions.h"
00011
00012 #include <vector>
00013
00014 class OrFilter : public FilterObject {
00015 public:
00016   std::vector<FilterObject *> filters;
00017   filterTypes getFilterType();
00018   ~OrFilter();
00019 };
00020
00021 #endif
```

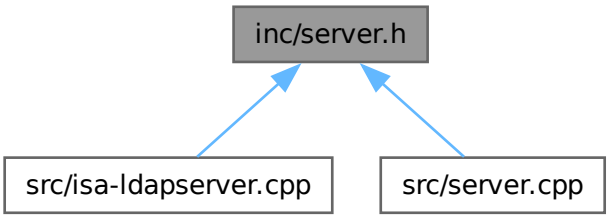
5.45 inc/server.h File Reference

ldap server implementation

```
#include "inc/BerEnumObject.h"
#include "inc/BerIntObject.h"
#include "inc/BerObject.h"
#include "inc/BerParser.h"
#include "inc/BerSequenceObject.h"
#include "inc/BerSetObject.h"
#include "inc/BerStringObject.h"
#include "inc/DatabaseObject.h"
#include "inc/FilterObject.h"
#include "inc/argument_helper_functions.h"
#include "inc/ldap_comunication.h"
#include <arpa/inet.h>
#include <fcntl.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/resource.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
```

Include dependency graph for server.h:

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



Macros

- #define CHECK_ERR(err, msg)

Functions

- int ldapServer (int port, char *dbPath)

  Ldap server, This part was inspired by the example from stubs demo tcp server https://git.fit.vutbr.cz/NESFIT/IPK-Projekty/src/branch/master/Stubs/cpp/DemoTcp by Vladimir Vesely Ph.D.

5.45.1 Detailed Description

ldap server implementation

Author

your name (you@domain.com)

Version

0.1

Date

2023-11-19

Copyright

Copyright (c) 2023

Definition in file server.h.

5.45.2 Macro Definition Documentation

5.45.2.1 CHECK_ERR

```
#define CHECK_ERR(
                err,
                msg )
```

Value:

```
  if (err != 0) {                                                    \
    printf("%s\n", msg);                                             \
    close(childSocket);                                              \
    exit(0);                                                         \
  }
```

Definition at line 38 of file server.h.

5.45.3 Function Documentation

5.45.3.1 ldapServer()

```
int ldapServer (
              int port,
              char * dbPath )
```

Ldap server, This part was inspired by the example from stubs demo tcp server https://git.fit.vutbr.cz/NESFIT/IPK-Projekty/src/branch/master/Stubs/cpp/DemoTcp by Vladimir Vesely Ph.D.

Parameters

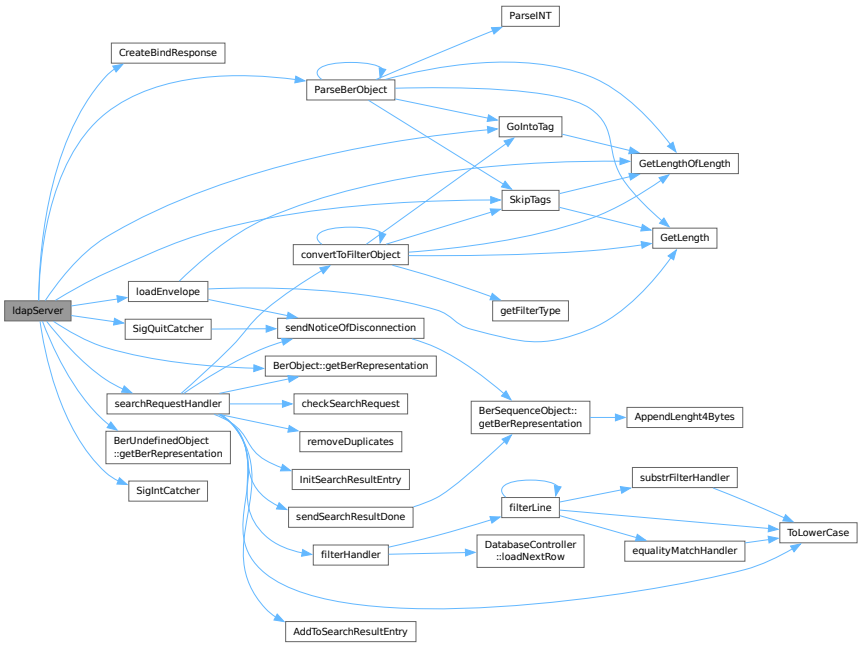
port	port to listen on
dbPath	path to database file

Returns

int

Definition at line 60 of file server.cpp.

Here is the call graph for this function:



5.46 server.h

Go to the documentation of this file.

```
00001
00011 #ifndef SERVER_H
00012 #define SERVER_H
00013 #include "inc/BerEnumObject.h"
00014 #include "inc/BerIntObject.h"
00015 #include "inc/BerObject.h"
00016 #include "inc/BerParser.h"
00017 #include "inc/BerSequenceObject.h"
00018 #include "inc/BerSetObject.h"
00019 #include "inc/BerStringObject.h"
00020 #include "inc/DatabaseObject.h"
00021 #include "inc/FilterObject.h"
00022 #include "inc/argument_helper_functions.h"
00023 #include "inc/ldap_comunication.h"
00024 #include <arpa/inet.h>
00025 #include <fcntl.h>
00026 #include <netinet/in.h>
00027 #include <stdio.h>
00028 #include <stdlib.h>
00029 #include <string.h>
```

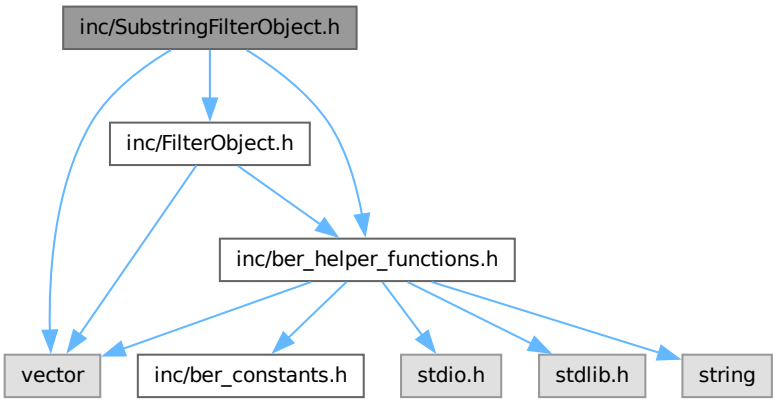
```
00030 #include <sys/resource.h>
00031 #include <sys/socket.h>
00032 #include <sys/time.h>
00033 #include <sys/types.h>
00034 #include <sys/wait.h>
00035 #include <unistd.h>
00036
00037 // Macro for printing err message and closing socket when err != 0
00038 #define CHECK_ERR(err, msg)                                                \
00039   if (err != 0) {                                                          \
00040     printf("%s\n", msg);                                                   \
00041     close(childSocket);                                                    \
00042     exit(0);                                                               \
00043   }
00044
00045 int ldapServer(int port, char *dbPath);
00046
00047 #endif
```

5.47 inc/SubstringFilterObject.h File Reference

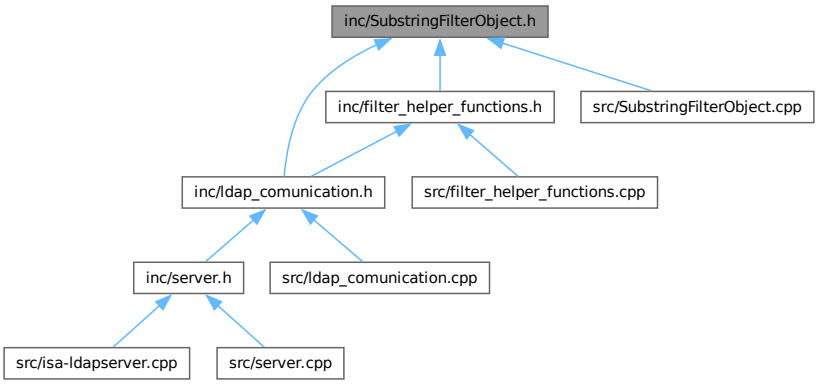
Object for substring filter.

```
#include "inc/ber_helper_functions.h"
#include "inc/FilterObject.h"
#include <vector>
```

Include dependency graph for SubstringFilterObject.h:



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



Classes

- class SubstringFilter

5.47.1 Detailed Description

Object for substring filter.

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file SubstringFilterObject.h.

5.48 SubstringFilterObject.h

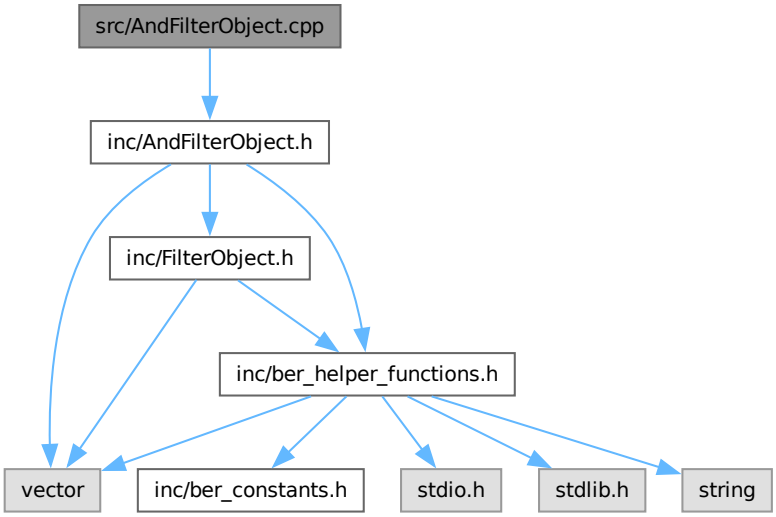
Go to the documentation of this file.

```
00001
00007 #ifndef SUBSTRINGFILTER_H
00008 #define SUBSTRINGFILTER_H
00009 #include "inc/ber_helper_functions.h"
00010 #include "inc/FilterObject.h"
00011 #include <vector>
00012
00013 class SubstringFilter : public FilterObject {
00014
00015 private:
00016   std::vector<unsigned char> attributeDescription;
00017   std::vector<unsigned char> subInitial;
00018   std::vector<std::vector<unsigned char» subAny;
00019   std::vector<unsigned char> subFinal;
00020
00021 public:
```

```
00022   SubstringFilter(std::vector<unsigned char> attributeDescription,
00023                   std::vector<unsigned char> subInitial,
00024                   std::vector<std::vector<unsigned char>> subAny,
00025                   std::vector<unsigned char> subFinal);
00026   std::vector<unsigned char> getAttributeDescription();
00027   std::vector<unsigned char> getSubInitial();
00028   std::vector<std::vector<unsigned char>> getSubAny();
00029   std::vector<unsigned char> getSubFinal();
00030   filterTypes getFilterType();
00031 };
00032
00033 #endif
```

5.49 src/AndFilterObject.cpp File Reference

`#include "inc/AndFilterObject.h"`
Include dependency graph for AndFilterObject.cpp:



5.49.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file AndFilterObject.cpp.

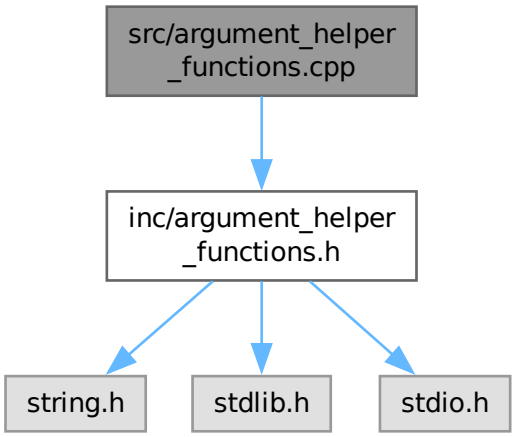
5.50 AndFilterObject.cpp

Go to the documentation of this file.

```
00001
00007 #include "inc/AndFilterObject.h"
00008
00009 filterTypes AndFilter::getFilterType() { return AND; };
00010 AndFilter::~AndFilter() {
00011     for (std::vector<FilterObject *>::iterator it = filters.begin();
00012          it != filters.end(); ++it) {
00013         delete *it;
00014     }
00015 }
```

5.51 src/argument_helper_functions.cpp File Reference

`#include "inc/argument_helper_functions.h"`
Include dependency graph for argument_helper_functions.cpp:



Functions

- argsT parseArguments (int argc, const char **argv)

  Parses the arguments from the command line for ldapserver.

5.51.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file argument_helper_functions.cpp.

5.51.2 Function Documentation

5.51.2.1 parseArguments()

```
argsT parseArguments (
            int argc,
            const char ** argv )
```

Parses the arguments from the command line for ldapserver.

Parameters

argc	count of arguments
argv	values of arguments

Returns

argsT

Definition at line 8 of file argument_helper_functions.cpp.

5.52 argument_helper_functions.cpp

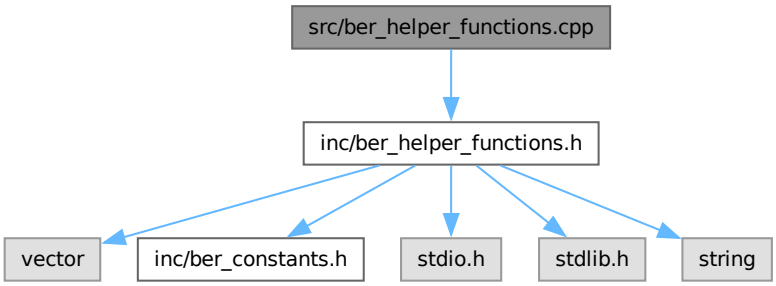
Go to the documentation of this file.

```
00001
00006 #include "inc/argument_helper_functions.h"
00007
00008 argsT parseArguments(int argc, const char **argv) {
00009
00010   // initialize args
00011   argsT args;
00012   args.err = false;
00013   args.dbPath = (char *)malloc(sizeof(char) * 1000);
00014   args.port = 389;
00015
00016   // main loop for parsing arguments
00017   for (int i = 1; i < argc; i++) {
00018     if (strcmp(argv[i], "-f") == 0) {
00019       // check if there is value for path to file
00020       if (i + 1 >= argc) {
00021         strcpy(args.dbPath, "");
00022         return args;
00023       } else {
00024         strcpy(args.dbPath, argv[i + 1]);
00025       }
00026       i++; // skip next argument
00027       continue;
00028     }
00029     if (strcmp(argv[i], "-p") == 0) {
00030       // check if there is value for port and if it is in range
00031       if (i + 1 >= argc || atoi(argv[i + 1]) <= 0 ||
00032           atoi(argv[i + 1]) > 65535) {
00033         args.err = true;
00034         return args;
00035       }
00036       args.port = atoi(argv[i + 1]);
00037       i++; // skip next argument
00038       continue;
00039     }
00040     // check if there is unknown argument
00041     fprintf(stderr, "Unknown argument: %s\n", argv[i]);
00042     args.err = true;
00043     return args;
00044   }
00045   return args;
00046 }
```

5.53 src/ber_helper_functions.cpp File Reference

```
#include "inc/ber_helper_functions.h"
```

Include dependency graph for ber_helper_functions.cpp:



Functions

- filterTypes getFilterType (std::vector< unsigned char >::iterator start)

  Parses type of filter from char array and returns its enum.
- unsigned int ParseINT (std::vector< unsigned char >::iterator s, int *err, std::vector< unsigned char >::iterator end)

  Parses value of BERInteger from char array.
- int GetLengthOfLength (std::vector< unsigned char >::iterator start, int *err, std::vector< unsigned char >::iterator end)

  Get the Length Of Length of BER attribute.
- std::vector< unsigned char > ToLowerCase (std::vector< unsigned char > input)

  converts std::vector<unsigned char> to lowercase
- int GetLength (std::vector< unsigned char >::iterator start, int *err, std::vector< unsigned char >::iterator end)

  Get the length of the data of BER attribute.
- void SkipTags (std::vector< unsigned char >::iterator &start, int n, int *err, std::vector< unsigned char >::iterator end)

  skips n BER attributes from char array, returns incremented iterator
- void IncreaseLength4Bytes (std::vector< unsigned char >::iterator &start, int n, int *err, std::vector< unsigned char >::iterator end)

  Increases 4Bytes longform length of the attribute by n.
- void AppendLenght4Bytes (std::vector< unsigned char > &start, int value)

  appends length in BER LDAP format to char array (4 bytes)
- int WriteIntAppend (std::vector< unsigned char > &s, int value)

  writes int in BER LDAP format to char array
- int HowManyBytesWillIntUse (int value)

  returns the number of bytes that will be used to encode the int
- void GoIntoTag (std::vector< unsigned char >::iterator &start, int *err, std::vector< unsigned char >::iterator end)

  goes into the sequence/set tag and returns incremented iterator which points to the first attribute in the sequence/set

5.53.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file ber_helper_functions.cpp.

5.53.2 Function Documentation

5.53.2.1 AppendLenght4Bytes()

```
void AppendLenght4Bytes (
            std::vector< unsigned char > & start,
            int value )
```

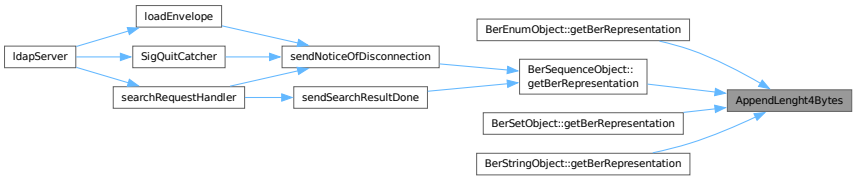
appends length in BER LDAP format to char array (4 bytes)

Parameters

start	vector to append to
value	
err	

Definition at line 161 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.53.2.2 getFilterType()

```
filterTypes getFilterType (
            std::vector< unsigned char >::iterator start )
```

Parses type of filter from char array and returns its enum.

Parameters

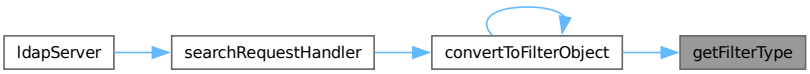
start	start of the filter

Returns

filterTypes

Definition at line 8 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.53.2.3 GetLength()

```
int GetLength (
            std::vector< unsigned char >::iterator start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Get the length of the data of BER attribute.

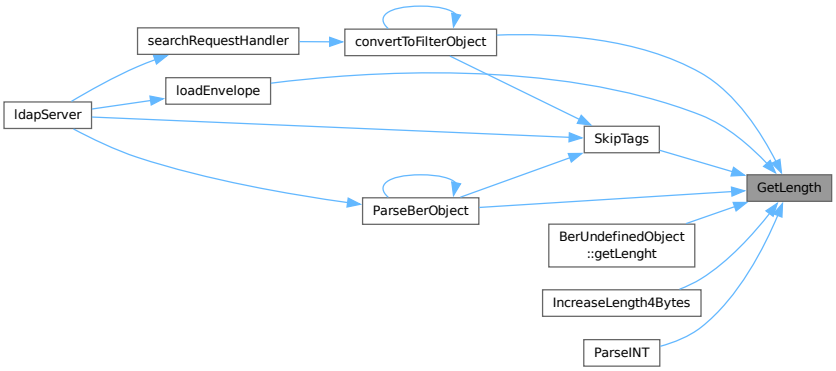
Parameters

start	start of the length
err	1 if error, 0 if success
end	end of the array

Returns

Definition at line 90 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.53.2.4 GetLengthOfLength()

```
int GetLengthOfLength (
            std::vector< unsigned char >::iterator start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Get the Length Of Length of BER attribute.

Parameters

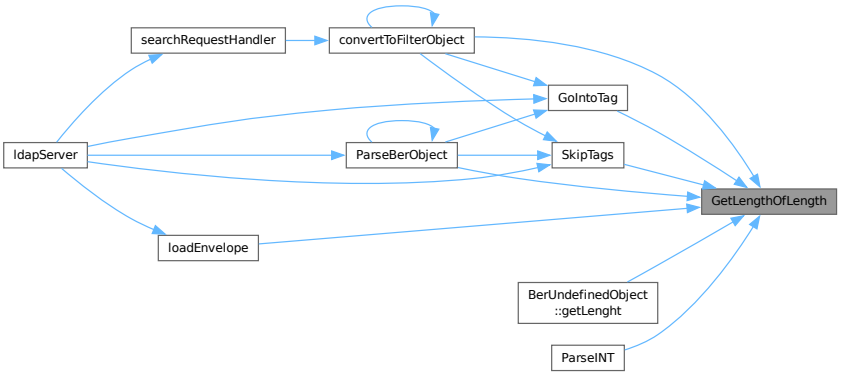
start	start of the length
err	1 if error, 0 if success
end	end of the array

Returns

int

Definition at line 63 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.53.2.5 GoIntoTag()

```
void GoIntoTag (
            std::vector< unsigned char >::iterator & start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

goes into the sequence/set tag and returns incremented iterator which points to the first attribute in the sequence/set

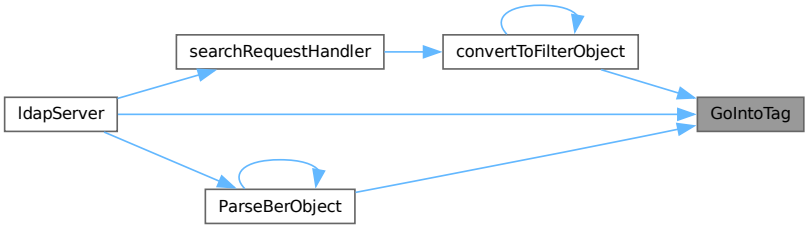
Parameters

start	start of the tag
err	1 if error, 0 if success
end	end of the array

Definition at line 254 of file ber_helper_functions.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



5.53.2.6 HowManyBytesWillIntUse()

```
int HowManyBytesWillIntUse (
            int value )
```

returns the number of bytes that will be used to encode the int

Parameters

value	

Returns

int

Definition at line 219 of file ber_helper_functions.cpp.

Here is the caller graph for this function:

5.53.2.7 IncreaseLength4Bytes()

```
void IncreaseLength4Bytes (
            std::vector< unsigned char >::iterator & start,
            int n,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Increases 4Bytes longform length of the attribute by n.

Parameters

start	start of the length
n	number by which the length will be increased
err	1 if error, 0 if success
end	end of the array

Definition at line 150 of file ber_helper_functions.cpp.

Here is the call graph for this function:

5.53.2.8 ParseINT()

```
unsigned int ParseINT (
             std::vector< unsigned char >::iterator s,
             int * err,
             std::vector< unsigned char >::iterator end )
```

Parses value of BERInteger from char array.

Parameters

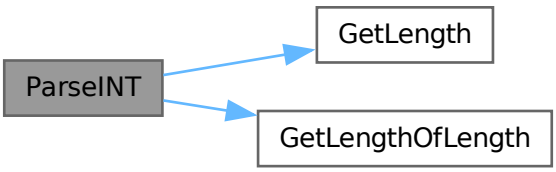
s	start of the integer in char array
err	1 if error, 0 if success
end	end of the array

Returns

unsigned int

Definition at line 23 of file ber_helper_functions.cpp.

Here is the call graph for this function:



5.53.2.9 SkipTags()

```
void SkipTags (
            std::vector< unsigned char >::iterator & start,
            int n,
            int * err,
            std::vector< unsigned char >::iterator end )
```

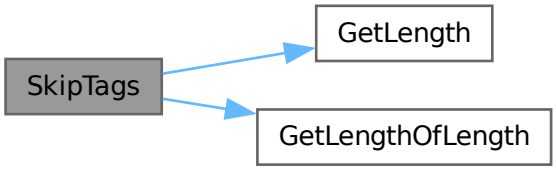
skips n BER attributes from char array, returns incremented iterator

Parameters

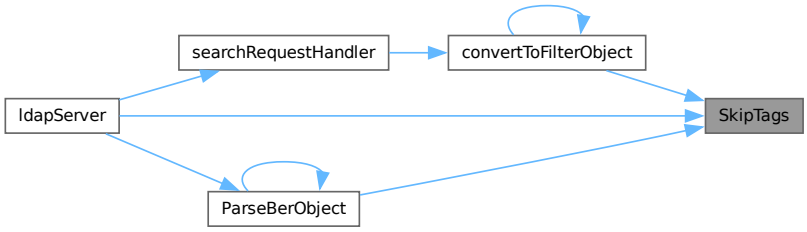
start	start of the tag
n	number of tags to skip
err	1 if error, 0 if success
end	end of the array

Definition at line 126 of file ber_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.53.2.10 ToLowerCase()

```
std::vector< unsigned char > ToLowerCase (
            std::vector< unsigned char > input )
```

converts std::vector<unsigned char> to lowercase

Parameters

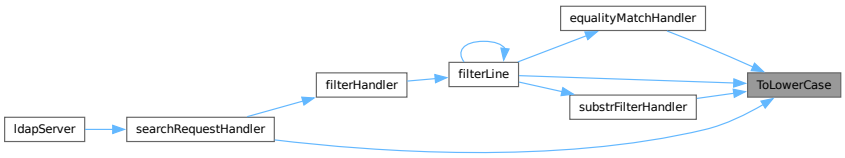
input	std::vector<unsigned char> to be converted

Returns

converted std::vector<unsigned char>

Definition at line 82 of file ber_helper_functions.cpp.

Here is the caller graph for this function:



5.53.2.11 WriteIntAppend()

```
int WriteIntAppend (
            std::vector< unsigned char > & s,
            int value )
```

writes int in BER LDAP format to char array

Parameters

s	start of the string in char array
value	int to be written

Returns

-1 if error, 0 if success

Definition at line 169 of file ber_helper_functions.cpp.

Here is the caller graph for this function:

5.54 ber_helper_functions.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/ber_helper_functions.h"
00007
00008 filterTypes getFilterType(std::vector<unsigned char>::iterator start) {
00009   if (start[0] == 0xA0) {
00010     return AND;
00011   } else if (start[0] == 0xA1) {
00012     return OR;
```

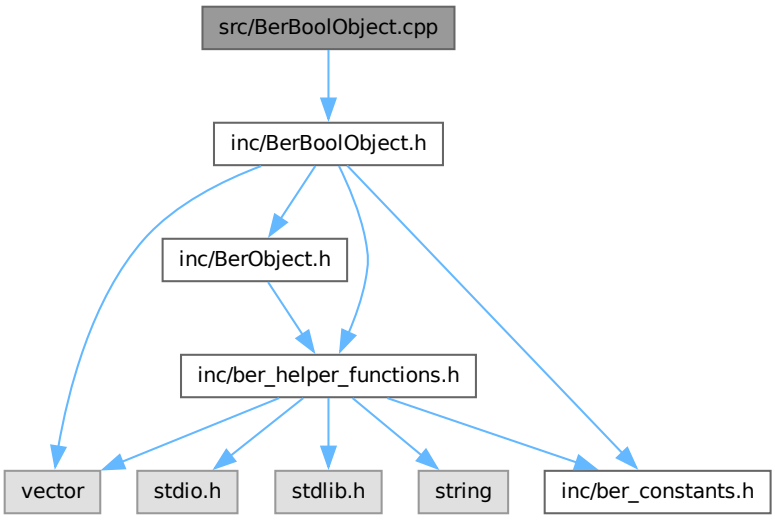
```
00013   } else if (start[0] == 0xA2) {
00014     return NOT;
00015   } else if (start[0] == 0xA3) {
00016     return equalityMatch;
00017   } else if (start[0] == 0xA4) {
00018     return substrings;
00019   }
00020   return undefined;
00021 }
00022
00023 unsigned int ParseINT(std::vector<unsigned char>::iterator s, int *err,
00024                       std::vector<unsigned char>::iterator end) {
00025   unsigned int value = 0;
00026   int length = GetLength(s + 1, err, end);
00027   int lengthLength = GetLengthOfLength(s + 1, err, end);
00028   if (length > 4) {
00029     return 0;
00030     *err = 2;
00031   }
00032
00033   switch (length) {
00034   case 0:
00035     value = 0;
00036     *err = 1;
00037     break;
00038   case 1:
00039     value = s[1 + lengthLength];
00040     err = 0;
00041     break;
00042   case 2:
00043     value = s[1 + lengthLength] << 8 | s[2 + lengthLength];
00044     err = 0;
00045     break;
00046   case 3:
00047     value = s[1 + lengthLength] << 16 | s[2 + lengthLength] << 8 |
00048             s[3 + lengthLength];
00049     err = 0;
00050     break;
00051   case 4:
00052     value = s[1 + lengthLength] << 24 | s[2 + lengthLength] << 16 |
00053             s[3 + lengthLength] << 8 | s[4 + lengthLength];
00054     err = 0;
00055     break;
00056
00057   default:
00058     break;
00059   }
00060   return value;
00061 }
00062
00063 int GetLengthOfLength(std::vector<unsigned char>::iterator start, int *err,
00064                       std::vector<unsigned char>::iterator end) {
00065
00066   if (std::distance(start, end) < 1) {
00067     *err = 1;
00068     return 0;
00069   }
00070
00071   int length = 0;
00072   if ((start[0] >> 7) != 1) { // if first bit is 0 -> shortform
00073     length = 1;
00074     *err = 0;
00075   } else {
00076     length = start[0] & 0x7F;
00077     length += 1;
00078   }
00079   return length;
00080 }
00081
00082 std::vector<unsigned char> ToLowerCase(std::vector<unsigned char> input) {
00083   std::vector<unsigned char> result;
00084   for (unsigned long int i = 0; i < input.size(); i++) {
00085     result.push_back(std::tolower(input[i]));
00086   }
00087   return result;
00088 }
00089
00090 int GetLength(std::vector<unsigned char>::iterator start, int *err,
00091               std::vector<unsigned char>::iterator end) {
00092   if (std::distance(start, end) < 1) {
00093     *err = 1;
00094     return 0;
00095   }
00096
00097   int length = 0;
00098   if ((start[0] >> 7) != 1) { // if first bit is 0 -> shortform
00099     length = start[0];
```

```
00100     *err = 0;
00101   } else {
00102     int lengthOfLength = start[0] & 0x7F; // remove bit indicating longform
00103
00104     if (lengthOfLength > std::distance(start, end)) { // array is too short
00105       *err = 1;
00106       return 0;
00107     }
00108     if (lengthOfLength > 4 &&
00109         start[lengthOfLength - 4] > 0x00) { // only support up to 4 bytes, more
00110                                             // is not necessary for this project
00111       *err = 1;
00112       return 0;
00113     }
00114     int startOfLength = lengthOfLength - 3;
00115     if (startOfLength < 0) {
00116       startOfLength = 1;
00117     }
00118     for (int i = startOfLength; i <= lengthOfLength; i++) {
00119       length = length « 8 | start[i];
00120     }
00121     *err = 0;
00122   }
00123   return length;
00124 }
00125
00126 void SkipTags(std::vector<unsigned char>::iterator &start, int n, int *err,
00127               std::vector<unsigned char>::iterator end) {
00128
00129   int i = 0;
00130   int jumpLength = 1;
00131   while (i < n) {
00132     if (std::distance(start, end) < jumpLength) {
00133       *err = 1;
00134       return;
00135     }
00136     int length =
00137         GetLength(start + jumpLength, err, end) +
00138         GetLengthOfLength(start + jumpLength, err, end);
00139     if (*err != 0) {
00140       *err = 1;
00141       return;
00142     }
00143
00144     jumpLength += length + 1; // +1 for tag
00145     i++;
00146   }
00147   start = start + jumpLength - 1; // -1 to get index of tag instead of length
00148 }
00149
00150 void IncreaseLength4Bytes(std::vector<unsigned char>::iterator &start, int n,
00151                           int *err, std::vector<unsigned char>::iterator end) {
00152
00153   int length = GetLength(start, err, end) + n;
00154
00155   start[1] = length » 24;
00156   start[2] = length » 16;
00157   start[3] = length » 8;
00158   start[4] = length;
00159 }
00160
00161 void AppendLenght4Bytes(std::vector<unsigned char> &start, int value) {
00162   start.push_back(0x84); // size
00163   start.push_back(value » 24);
00164   start.push_back(value » 16);
00165   start.push_back(value » 8);
00166   start.push_back(value);
00167 }
00168
00169 int WriteIntAppend(std::vector<unsigned char> &s, int value) {
00170   if (value < 0) {
00171     return -1;
00172   }
00173   int overflow = 0;
00174   s.push_back(BER_INT_C);
00175   if (value < 0x100) {
00176     s.push_back(0x01); // length
00177     if ((value) == 0xFF) {
00178       s.push_back(0x00);
00179       overflow = 1;
00180     }
00181     s.push_back(value);
00182     return 3 + overflow;
00183   } else if (value < 0x10000) {
00184     s.push_back(0x02); // length
00185     if ((value » 8) == 0xFF) {
00186       s.push_back(0x00);
```

```
00187       overflow = 1;
00188     }
00189     s.push_back(value » 8);
00190     s.push_back(value);
00191     return 4 + overflow;
00192   } else if (value < 0x1000000) {
00193     s.push_back(0x03); // length
00194     if ((value » 16) == 0xFF) {
00195       s.push_back(0x00);
00196       overflow = 1;
00197     }
00198     s.push_back(value » 16);
00199     s.push_back(value » 8);
00200     s.push_back(value);
00201     return 5 + overflow;
00202   } else if (value < 0x100000000) {
00203     s.push_back(0x04); // length
00204     if ((value » 24) == 0xFF) {
00205       s.push_back(0x00);
00206       overflow = 1;
00207     }
00208     s.push_back(value » 24);
00209     s.push_back(value » 16);
00210     s.push_back(value » 8);
00211     s.push_back(value);
00212     return 6 + overflow;
00213   } else {
00214     return -1;
00215   }
00216   return -1;
00217 }
00218
00219 int HowManyBytesWillIntUse(int value) {
00220   if (value < 0) {
00221     return -1;
00222   }
00223   if (value < 0x100) {
00224     if ((value) == 0xFF) {
00225       return 4;
00226     } else {
00227       return 3;
00228     }
00229
00230   } else if (value < 0x10000) {
00231     if ((value » 8) == 0xFF) {
00232       return 5;
00233     } else {
00234       return 4;
00235     }
00236   } else if (value < 0x1000000) {
00237     if ((value » 16) == 0xFF) {
00238       return 6;
00239     } else {
00240       return 5;
00241     }
00242   } else if (value < 0x100000000) {
00243     if ((value » 24) == 0xFF) {
00244       return 7;
00245     } else {
00246       return 6;
00247     }
00248   } else {
00249     return -1;
00250   }
00251   return -1;
00252 }
00253
00254 void GoIntoTag(std::vector<unsigned char>::iterator &start, int *err,
00255                std::vector<unsigned char>::iterator end) {
00256
00257   int length = GetLengthOfLength(start + 1, err, end) + 1; // +1 for tag
00258   if (*err != 0) {
00259     *err = 1;
00260     return;
00261   }
00262
00263   start += length; // return pointer to next tag
00264 }
```

5.55 src/BerBoolObject.cpp File Reference

`#include "inc/BerBoolObject.h"`
Include dependency graph for BerBoolObject.cpp:



5.55.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerBoolObject.cpp.

5.56 BerBoolObject.cpp

Go to the documentation of this file.

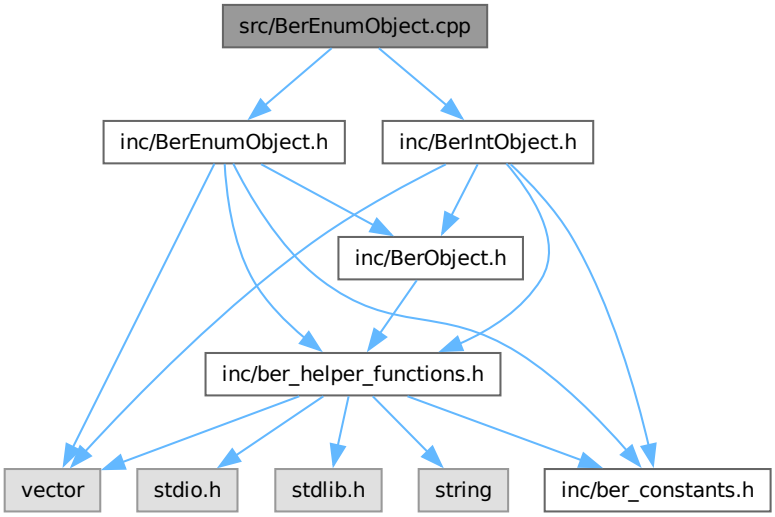
```
00001
00006 #include "inc/BerBoolObject.h"
00007
00008 berObjectTypes BerBoolObject::getBerObjectType() { return berBoolObject; }
00009
00010 long long int BerBoolObject::getLenght() {
00011   const int BOOL_DATA_LENGTH = 1;
00012   return BER_TAG_LENGTH + BER_LENGTH_OF_LENGTH_TAG + BOOL_DATA_LENGTH;
00013 }
00014
00015 std::vector<unsigned char> BerBoolObject::getBerRepresentation() {
00016   const int BOOL_DATA_LENGTH = 1;
00017   std::vector<unsigned char> berRepresentation;
00018   berRepresentation.push_back(BER_BOOL_C);
```

```
00019   berRepresentation.push_back(BOOL_DATA_LENGTH);
00020   if (value > 0) {
00021     berRepresentation.push_back(0xFF);
00022   } else {
00023     berRepresentation.push_back(0x00);
00024   }
00025   return berRepresentation;
00026 }
00027
00028 BerBoolObject::BerBoolObject(char value) { this->value = value; }
00029
00030 BerBoolObject::~BerBoolObject() {}
```

5.57 src/BerEnumObject.cpp File Reference

```
#include "inc/BerEnumObject.h"
#include "inc/BerIntObject.h"
```

Include dependency graph for BerEnumObject.cpp:



5.57.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerEnumObject.cpp.

5.58 BerEnumObject.cpp

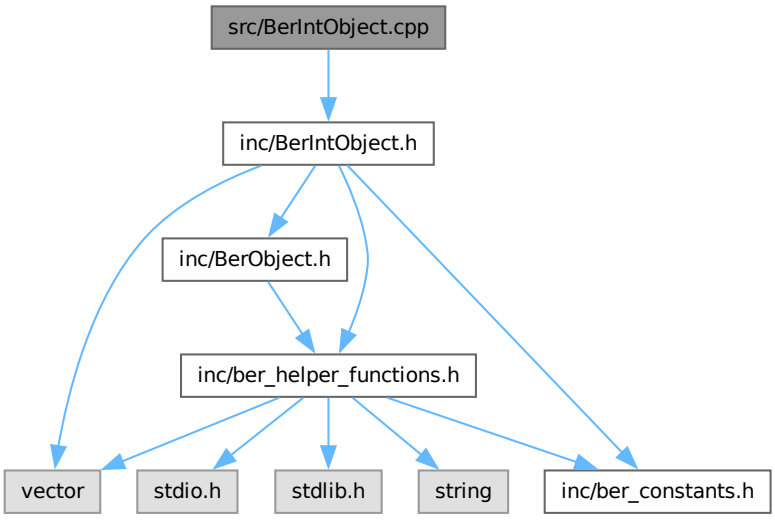
Go to the documentation of this file.

```
00001
00006 #include "inc/BerEnumObject.h"
00007
00008 #include "inc/BerIntObject.h"
00009
00010 berObjectTypes BerEnumObject::getBerObjectType() { return berEnumObject; }
00011
00012 long long int BerEnumObject::getLenght() {
00013   const int ENUM_DATA_LENGTH = 1;
00014   return BER_TAG_LENGTH + BER_LENGTH_OF_LENGTH_TAG + BER_4BYTE_LENGTH_LENGTH +
00015          ENUM_DATA_LENGTH;
00016 }
00017
00018 std::vector<unsigned char> BerEnumObject::getBerRepresentation() {
00019   const int ENUM_DATA_LENGTH = 1;
00020   std::vector<unsigned char> berRepresentation;
00021   berRepresentation.push_back(BER_ENUM_C);
00022   AppendLenght4Bytes(berRepresentation, ENUM_DATA_LENGTH);
00023   berRepresentation.push_back(value);
00024   return berRepresentation;
00025 }
00026
00027 BerEnumObject::BerEnumObject(char value) { this->value = value; }
00028
00029 BerEnumObject::~BerEnumObject() {}
```

5.59 src/BerIntObject.cpp File Reference

`#include "inc/BerIntObject.h"`

Include dependency graph for BerIntObject.cpp:



5.59.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerIntObject.cpp.

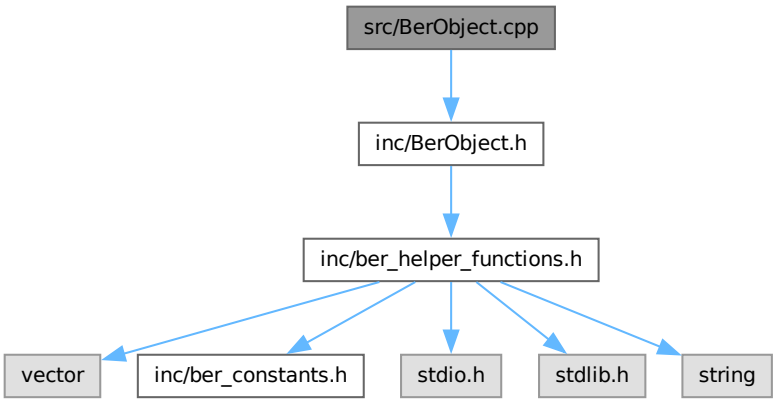
5.60 BerIntObject.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/BerIntObject.h"
00007
00008 berObjectTypes BerIntObject::getBerObjectType() { return berIntObject; }
00009
00010 long long int BerIntObject::getLenght() {
00011
00012   return HowManyBytesWillIntUse(value);
00013 }
00014
00015 std::vector<unsigned char> BerIntObject::getBerRepresentation() {
00016   std::vector<unsigned char> berRepresentation;
00017   WriteIntAppend(berRepresentation, value);
00018   return berRepresentation;
00019 }
00020
00021 BerIntObject::BerIntObject() { value = 0; }
00022 BerIntObject::BerIntObject(int value) { this->value = value; }
00023
00024 int BerIntObject::getValue() { return value; }
00025
00026 void BerIntObject::setValue(int value) { this->value = value; }
00027
00028 BerIntObject::~BerIntObject() {}
```

5.61 src/BerObject.cpp File Reference

`#include "inc/BerObject.h"`
Include dependency graph for BerObject.cpp:



5.61.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerObject.cpp.

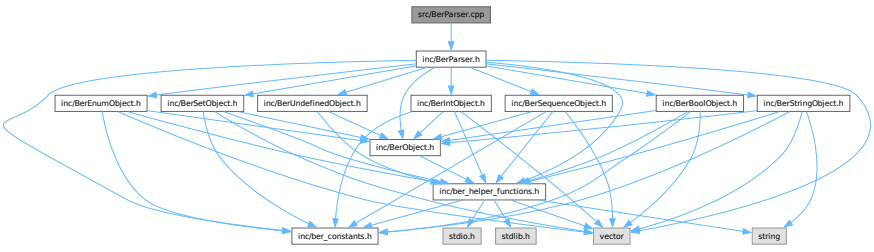
5.62 BerObject.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/BerObject.h"
00007
00008
00009 berObjectTypes BerObject::getBerObjectType(){
00010     return berErr;
00011 }
00012 long long int  BerObject::getLenght(){
00013     return -1;
00014 }
00015 std::vector<unsigned char> BerObject::getBerRepresentation(){
00016     std::vector<unsigned char> result;
00017     return result;
00018 }
00019
00020 BerObject::~BerObject(){
00021
00022 }
```

5.63 src/BerParser.cpp File Reference

`#include "inc/BerParser.h"`
Include dependency graph for BerParser.cpp:



Functions

- BerObject * ParseBerObject (std::vector< unsigned char >::iterator start, int *err, std::vector< unsigned char >::iterator end)

  Parses BER encoded message and converts it to BerObject, it works recursively.

5.63.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerParser.cpp.

5.63.2 Function Documentation

5.63.2.1 ParseBerObject()

```
BerObject * ParseBerObject (
            std::vector< unsigned char >::iterator start,
            int * err,
            std::vector< unsigned char >::iterator end )
```

Parses BER encoded message and converts it to BerObject, it works recursively.

Parses BER and converts it to BerObject.

Parameters

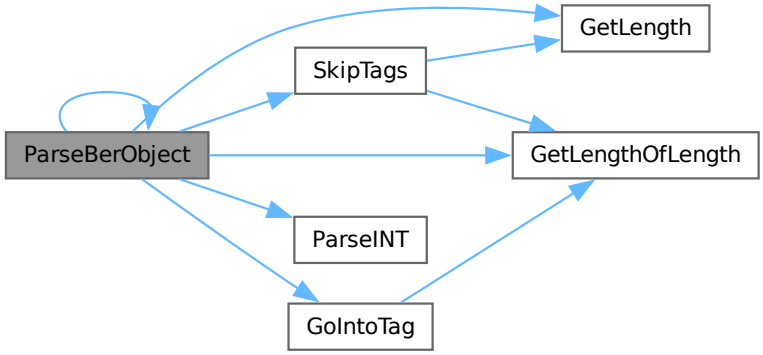
start	start of the BER
err	1 if error, 0 if success
end	end of the array

Returns

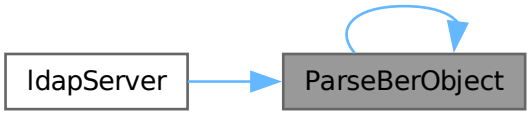
BerObject*

Definition at line 16 of file BerParser.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.64 BerParser.cpp

Go to the documentation of this file.

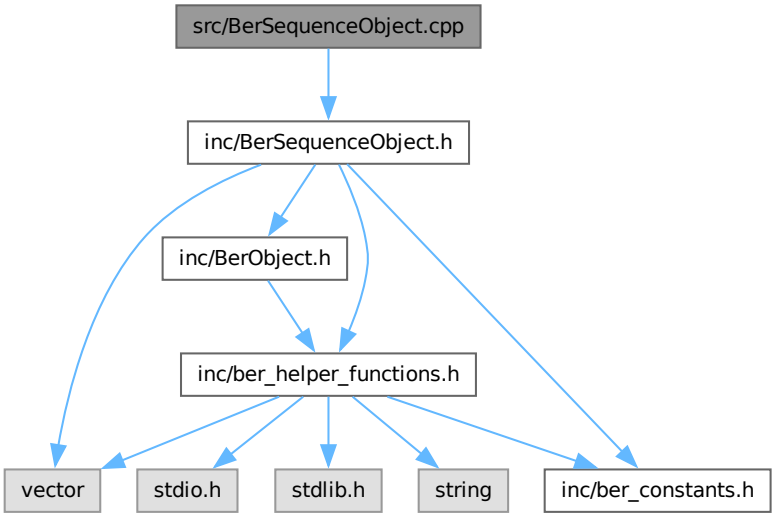
```
00001
00006 #include "inc/BerParser.h"
00007
00016 BerObject *ParseBerObject(std::vector<unsigned char>::iterator start,
00017                           int *err, std::vector<unsigned char>::iterator end) {
00018   BerObject *berObject;
00019
00020   unsigned char tag = start[0];
00021     // main switch for all types
00022   switch (tag) {
00023     {
00024     case BER_SEQUENCE_C:
00025       BerSequenceObject *berSequenceObject = new BerSequenceObject();
00026       int length = GetLength(start + 1, err, end);
00027       GoIntoTag(start, err, end);
00028       GetLengthOfLength(start +1, err, end);
00029       int i = 0;
00030       while (i < length) {
00031         BerObject *parsedBerObject = ParseBerObject(start, err, end);
00032         berSequenceObject->objects.push_back(parsedBerObject);
00033           i += BER_TAG_LENGTH + GetLengthOfLength(start +1, err, end) +
00034              GetLength(start + 1, err, end);
00035         SkipTags(start, 1, err, end);
00036       }
00037       berObject = berSequenceObject;
00038       break;
```

```
00039     }
00040     {
00041     case BER_BIND_REQUEST_C:
00042       BerSequenceObject *berSequenceObject =
00043           new BerSequenceObject(BER_BIND_REQUEST_C);
00044       int length = GetLength(start + 1, err, end);
00045       GoIntoTag(start, err, end);
00046
00047       GetLengthOfLength(start +1, err, end);
00048       int i = 0;
00049       while (i < length) {
00050         BerObject *parsedBerObject = ParseBerObject(start, err, end);
00051         berSequenceObject->objects.push_back(parsedBerObject);
00052           i += BER_TAG_LENGTH + GetLengthOfLength(start +1, err, end) +
00053              GetLength(start + 1, err, end);
00054         SkipTags(start, 1, err, end);
00055       }
00056       berObject = berSequenceObject;
00057       break;
00058     }
00059     {
00060     case BER_SEARCH_REQUEST_C:
00061       BerSequenceObject *berSequenceObject =
00062           new BerSequenceObject(BER_SEARCH_REQUEST_C);
00063       int length = GetLength(start + 1, err, end);
00064       GoIntoTag(start, err, end);
00065
00066       GetLengthOfLength(start +1, err, end);
00067       int i = 0;
00068       while (i < length) {
00069         BerObject *parsedBerObject = ParseBerObject(start, err, end);
00070         berSequenceObject->objects.push_back(parsedBerObject);
00071         i += BER_TAG_LENGTH + GetLengthOfLength(start +1, err, end) +
00072              GetLength(start + 1, err, end);
00073         SkipTags(start, 1, err, end);
00074       }
00075       berObject = berSequenceObject;
00076       break;
00077     }
00078   case BER_INT_C: {
00079     BerIntObject *berIntObject = new BerIntObject();
00080     int value = ParseINT(start, err, end);
00081     berIntObject->setValue(value);
00082     berObject = berIntObject;
00083     break;
00084   }
00085
00086   case BER_ENUM_C: {
00087     int lenghtOfLenght = GetLengthOfLength(start +1 , err, end);
00088
00089     char value = start[lenghtOfLenght + BER_TAG_LENGTH];
00090
00091     berObject = new BerEnumObject(value);
00092     break;
00093   }
00094   case BER_SET_C: {
00095     BerSetObject *berSetObject = new BerSetObject();
00096     int length = GetLength(start + 1, err, end);
00097     GoIntoTag(start, err,end);
00098
00099     int i = 0;
00100     while (i < length) {
00101
00102       BerObject *parsedBerObject = ParseBerObject(start, err, end);
00103       berSetObject->objects.push_back(parsedBerObject);
00104       i += BER_TAG_LENGTH + GetLengthOfLength(start +1, err, end) +
00105              GetLength(start + 1, err, end);
00106       SkipTags(start, 1, err, end);
00107     }
00108     berObject = berSetObject;
00109     break;
00110   }
00111   case BER_OCTET_STRING_C: {
00112
00113     int lenghtOfLenght = GetLengthOfLength(start +1, err, end);
00114     int length = GetLength(start + 1, err, end);
00115     BerStringObject *berStringObject = new BerStringObject();
00116     if (length == 0) {
00117       berStringObject->value = std::vector<unsigned char>();
00118       berObject = berStringObject;
00119       break;
00120     }
00121     berStringObject->value = std::vector<unsigned char>(
00122         start + lenghtOfLenght + BER_TAG_LENGTH,
00123         start + lenghtOfLenght + BER_TAG_LENGTH + length);
00124     berObject = berStringObject;
00125     break;
```

```
00126   }
00127   case BER_BOOL_C: {
00128     int lenghtOfLenght = GetLengthOfLength(start +1, err, end);
00129     GetLength(start + 1, err, end);
00130     berObject = new BerBoolObject(start[lenghtOfLenght + BER_TAG_LENGTH]);
00131     break;
00132   }
00133   default: // filters and other types
00134     std::vector<unsigned char> berVector = std::vector<unsigned char>(
00135         start, start + GetLengthOfLength(start+1, err, end) + BER_TAG_LENGTH +
00136                     GetLength(start + 1, err, end));
00137     berObject = new BerUndefinedObject(berVector);
00138
00139     break;
00140   }
00141   return berObject;
00142 }
```

5.65 src/BerSequenceObject.cpp File Reference

`#include "inc/BerSequenceObject.h"`
Include dependency graph for BerSequenceObject.cpp:



5.65.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerSequenceObject.cpp.

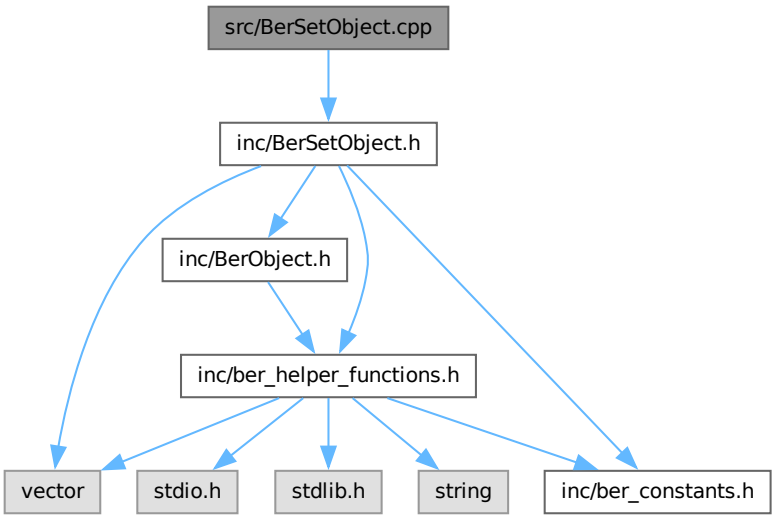
5.66 BerSequenceObject.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/BerSequenceObject.h"
00007
00008 berObjectTypes BerSequenceObject::getBerObjectType() {
00009   return berSequenceObject;
00010 }
00011
00012 long long int BerSequenceObject::getLenght() {
00013
00014   long long int dataLenght = 0;
00015
00016   for (long unsigned int i = 0; i < objects.size(); i++) {
00017     dataLenght += objects[i]->getLenght();
00018   }
00019
00020   return BER_TAG_LENGTH + BER_LENGTH_OF_LENGTH_TAG + BER_4BYTE_LENGTH_LENGTH +
00021          dataLenght;
00022 }
00023
00024 std::vector<unsigned char> BerSequenceObject::getBerRepresentation() {
00025   std::vector<unsigned char> berRepresentation;
00026   long long int dataLenght = 0;
00027   for (long unsigned int i = 0; i < objects.size(); i++) {
00028     dataLenght += objects[i]->getLenght();
00029   }
00030
00031   berRepresentation.push_back(tag);
00032   AppendLenght4Bytes(berRepresentation, dataLenght);
00033   for (long unsigned int i = 0; i < objects.size(); i++) {
00034     std::vector<unsigned char> objectRepresentation =
00035         objects[i]->getBerRepresentation();
00036     berRepresentation.insert(berRepresentation.end(),
00037                              objectRepresentation.begin(),
00038                              objectRepresentation.end());
00039   }
00040
00041   return berRepresentation;
00042 }
00043
00044 BerSequenceObject::BerSequenceObject() { tag = BER_SEQUENCE_C; }
00045
00046 int BerSequenceObject::GetTag() { return tag; }
00047
00048 BerSequenceObject::BerSequenceObject(int tag) { this->tag = tag; }
00049
00050 BerSequenceObject::~BerSequenceObject() {
00051   for (long unsigned int i = 0; i < objects.size(); i++) {
00052     delete objects[i];
00053   }
00054 }
```

5.67 src/BerSetObject.cpp File Reference

`#include "inc/BerSetObject.h"`
Include dependency graph for BerSetObject.cpp:



5.67.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerSetObject.cpp.

5.68 BerSetObject.cpp

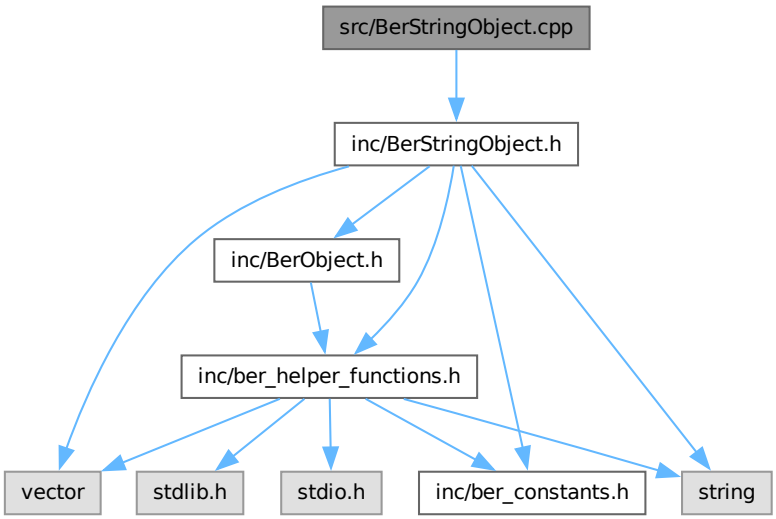
Go to the documentation of this file.

```
00001
00006 #include "inc/BerSetObject.h"
00007
00008 berObjectTypes BerSetObject::getBerObjectType() { return berSequenceObject; }
00009
00010 long long int BerSetObject::getLenght() {
00011
00012   long long int dataLenght = 0;
00013
00014   for (long unsigned int i = 0; i < objects.size(); i++) {
00015     dataLenght += objects[i]->getLenght();
00016   }
00017
00018   return BER_TAG_LENGTH + BER_LENGTH_OF_LENGTH_TAG + BER_4BYTE_LENGTH_LENGTH +
```

```
00019          dataLenght;
00020 }
00021
00022 std::vector<unsigned char> BerSetObject::getBerRepresentation() {
00023   std::vector<unsigned char> berRepresentation;
00024   long long int dataLenght = 0;
00025   for (long unsigned int i = 0; i < objects.size(); i++) {
00026     dataLenght += objects[i]->getLenght();
00027   }
00028
00029   berRepresentation.push_back(BER_SET_C);
00030   AppendLenght4Bytes(berRepresentation, dataLenght);
00031   for (long unsigned int i = 0; i < objects.size(); i++) {
00032     std::vector<unsigned char> objectRepresentation =
00033         objects[i]->getBerRepresentation();
00034     berRepresentation.insert(berRepresentation.end(),
00035                              objectRepresentation.begin(),
00036                              objectRepresentation.end());
00037   }
00038
00039   return berRepresentation;
00040 }
00041
00042 BerSetObject::BerSetObject() {}
00043
00044 BerSetObject::~BerSetObject() {
00045   for (long unsigned int i = 0; i < objects.size(); i++) {
00046     delete objects[i];
00047   }
00048 }
```

5.69 src/BerStringObject.cpp File Reference

`#include "inc/BerStringObject.h"`
Include dependency graph for BerStringObject.cpp:



5.69.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerStringObject.cpp.

5.70 BerStringObject.cpp

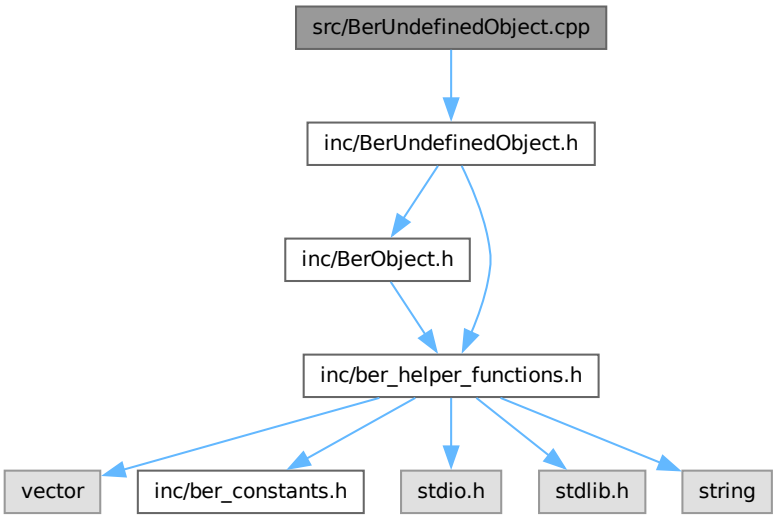
Go to the documentation of this file.

```
00001
00006 #include "inc/BerStringObject.h"
00007
00008 berObjectTypes BerStringObject::getBerObjectType() { return berStringObject; }
00009 long long int BerStringObject::getLenght() {
00010
00011   return BER_TAG_LENGTH + BER_LENGTH_OF_LENGTH_TAG + BER_4BYTE_LENGTH_LENGTH +
00012          value.size();
00013 }
00014
00015 std::vector<unsigned char> BerStringObject::getBerRepresentation() {
00016   std::vector<unsigned char> berRepresentation;
00017   berRepresentation.push_back(BER_OCTET_STRING_C);
00018   AppendLenght4Bytes(berRepresentation, value.size());
00019   berRepresentation.insert(berRepresentation.end(), value.begin(), value.end());
00020   return berRepresentation;
00021 }
00022
00023 BerStringObject::BerStringObject() {}
00024
00025 BerStringObject::BerStringObject(std::vector<unsigned char> value) {
00026   this->value = value;
00027 }
00028 BerStringObject::BerStringObject(std::string value) {
00029
00030   this->value = std::vector<unsigned char>(value.begin(), value.end());
00031 }
```

5.71 src/BerUndefinedObject.cpp File Reference

`#include "inc/BerUndefinedObject.h"`

Include dependency graph for BerUndefinedObject.cpp:



5.71.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file BerUndefinedObject.cpp.

5.72 BerUndefinedObject.cpp

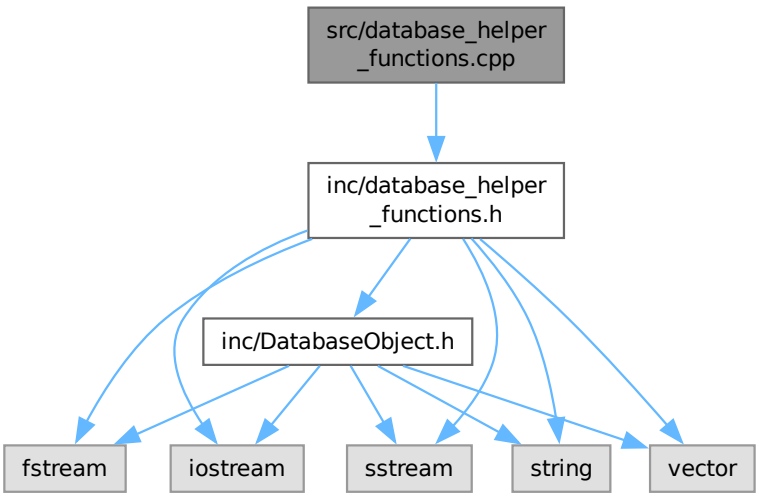
Go to the documentation of this file.

```
00001
00006 #include "inc/BerUndefinedObject.h"
00007
00008
00009 berObjectTypes BerUndefinedObject::getBerObjectType(){
00010     return berUndefined;
00011 }
00012 long long int  BerUndefinedObject::getLenght(){
00013     int err = 0;
00014     return BER_TAG_LENGTH + GetLengthOfLength(value.begin() +1, &err, value.end())  +
      GetLength(value.begin() +1, &err, value.end());
00015 }
00016 std::vector<unsigned char> BerUndefinedObject::getBerRepresentation(){
00017     return value;
00018 }
00019
00020 BerUndefinedObject::BerUndefinedObject(std::vector<unsigned char> value){
00021     this->value = value;
00022 }
```

5.73 src/database_helper_functions.cpp File Reference

`#include "inc/database_helper_functions.h"`

Include dependency graph for database_helper_functions.cpp:



Functions

- std::vector< DatabaseObject > removeDuplicates (std::vector< DatabaseObject > input)
  Removes dupplicates from vector of DatabaseObjects.

5.73.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file database_helper_functions.cpp.

5.73.2 Function Documentation

5.73.2.1 removeDuplicates()

```
std::vector< DatabaseObject > removeDuplicates (
            std::vector< DatabaseObject > input )
```

Removes dupplicates from vector of DatabaseObjects.

Parameters

input	

Returns

std::vector<DatabaseObject>

Definition at line 9 of file database_helper_functions.cpp.

Here is the caller graph for this function:

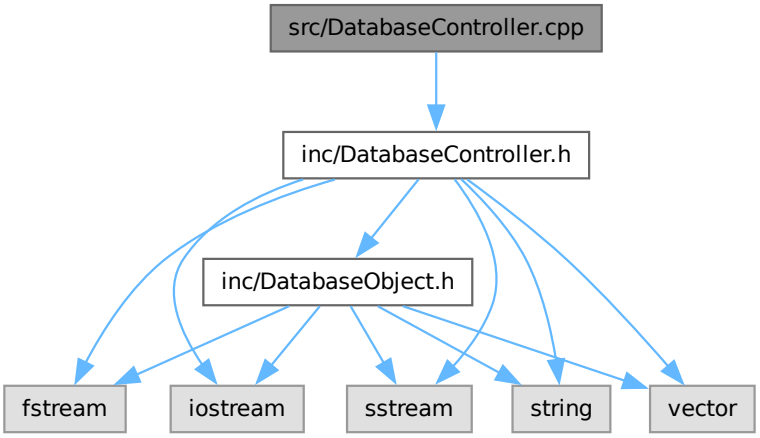
5.74 database_helper_functions.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/database_helper_functions.h"
00007
00008 std::vector<DatabaseObject>
00009 removeDuplicates(std::vector<DatabaseObject> input) {
00010   std::vector<DatabaseObject> result;
00011   for (unsigned long int i = 0; i < input.size(); i++) {
00012     bool found = false;
00013     for (unsigned long int j = 0; j < result.size(); j++) {
00014       if (input[i].get_uid() == result[j].get_uid()) {
00015         found = true;
00016         break;
00017       }
00018     }
00019     if (!found) {
00020       result.push_back(input[i]);
00021     }
00022   }
00023   return result;
00024 }
```

5.75 src/DatabaseController.cpp File Reference

`#include "inc/DatabaseController.h"`
Include dependency graph for DatabaseController.cpp:



5.75.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file DatabaseController.cpp.

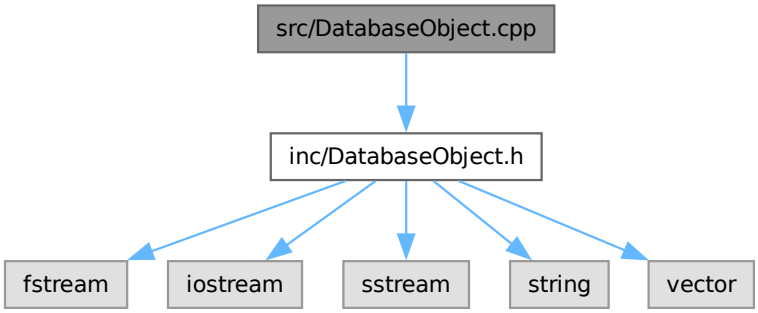
5.76 DatabaseController.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/DatabaseController.h"
00007
00008 std::vector<DatabaseObject> DatabaseController::loadAllRows() {
00009   std::vector<DatabaseObject> result;
00010   while (!file.eof()) {
00011     int err;
00012     result.push_back(loadNextRow(&err));
00013   }
00014   return result;
00015 }
00016
00017 std::vector<unsigned char>
00018 DatabaseController::sanitaze(std::vector<unsigned char> input) {
00019   std::vector<unsigned char> result;
00020   // allow only printable ascii charactes and numbers
00021   for (unsigned long int i = 0; i < input.size(); i++) {
00022     if (input[i] >= 32 && input[i] <= 126) {
00023       result.push_back(input[i]);
00024     }
00025   }
00026   return result;
00027 }
00028
00029 DatabaseObject DatabaseController::loadNextRow(int *err) {
00030
00031   if (file.eof()) {
00032     *err = 1;
00033     return DatabaseObject(std::vector<unsigned char>(),
00034                           std::vector<unsigned char>(),
00035                           std::vector<unsigned char>());
00036   }
00037
00038   std::vector<unsigned char> name;
00039   std::vector<unsigned char> uid;
00040   std::vector<unsigned char> email;
00041
00042   // read csv line
00043   std::string line;
00044   std::getline(file, line);
00045   if (line == "") {
00046     *err = 1;
00047     return DatabaseObject(std::vector<unsigned char>(),
00048                           std::vector<unsigned char>(),
00049                           std::vector<unsigned char>());
00050   }
00051   std::stringstream ss(line);
00052   std::string token;
00053   std::getline(ss, token, ';');
00054
00055   name = DatabaseController::sanitaze(
00056       std::vector<unsigned char>(token.begin(), token.end()));
00057   std::getline(ss, token, ';');
00058   uid = DatabaseController::sanitaze(
00059       std::vector<unsigned char>(token.begin(), token.end()));
00060   std::getline(ss, token, ';');
00061   email = DatabaseController::sanitaze(
00062       std::vector<unsigned char>(token.begin(), token.end()));
00063   *err = 0;
00064   return DatabaseObject(name, uid, email);
00065 }
00066
00067 DatabaseController::DatabaseController(std::string fileName) {
00068   file.open(fileName);
00069 }
00070
00071 DatabaseController::~DatabaseController() { file.close(); }
```

5.77 src/DatabaseObject.cpp File Reference

`#include "inc/DatabaseObject.h"`
Include dependency graph for DatabaseObject.cpp:



5.77.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file DatabaseObject.cpp.

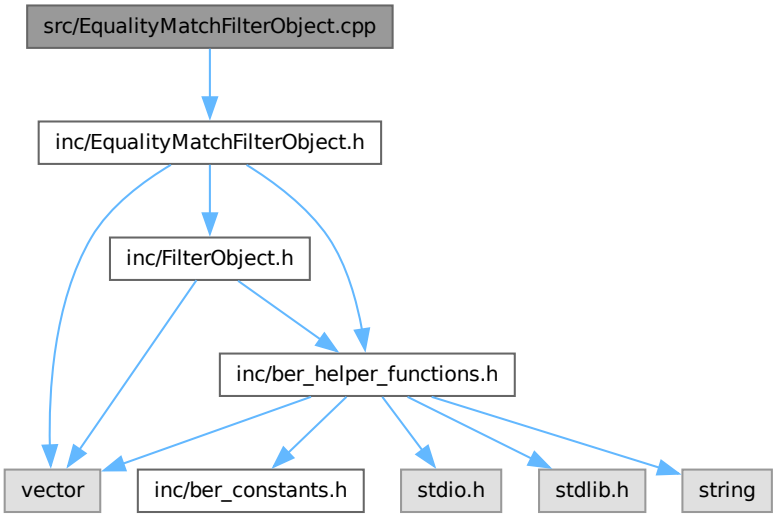
5.78 DatabaseObject.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/DatabaseObject.h"
00007
00008 std::vector<unsigned char> DatabaseObject::get_name() { return this->name; }
00009 std::vector<unsigned char> DatabaseObject::get_uid() { return this->uid; }
00010 std::vector<unsigned char> DatabaseObject::get_email() { return this->email; }
00011 DatabaseObject::DatabaseObject(std::vector<unsigned char> name,
00012                                std::vector<unsigned char> uid,
00013                                std::vector<unsigned char> email) {
00014   this->name = name;
00015   this->uid = uid;
00016   this->email = email;
00017 }
00018
00019
00020
00021
00022
00023
```

5.79 src/EqualityMatchFilterObject.cpp File Reference

`#include "inc/EqualityMatchFilterObject.h"`
Include dependency graph for EqualityMatchFilterObject.cpp:



5.79.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file EqualityMatchFilterObject.cpp.

5.80 EqualityMatchFilterObject.cpp

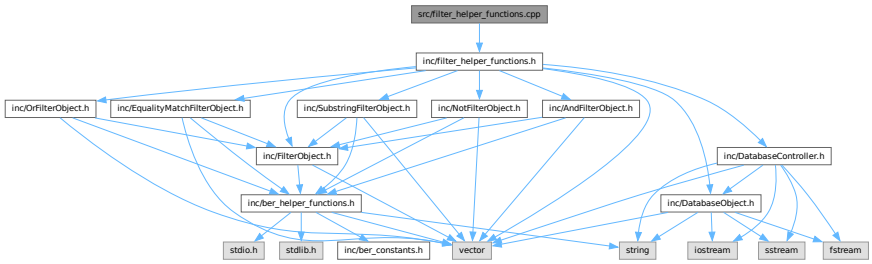
Go to the documentation of this file.

```
00001
00006 #include "inc/EqualityMatchFilterObject.h"
00007
00008 EqualityMatchFilter::EqualityMatchFilter(
00009     std::vector<unsigned char> attributeDescription,
00010     std::vector<unsigned char> assertionValue) {
00011   this->attributeDescription = attributeDescription;
00012   this->assertionValue = assertionValue;
00013 };
00014
00015 std::vector<unsigned char> EqualityMatchFilter::getAttributeDescription() {
00016   return attributeDescription;
00017 };
00018
00019 std::vector<unsigned char> EqualityMatchFilter::getAssertionValue() {
00020   return assertionValue;
00021 };
00022
00023 filterTypes EqualityMatchFilter::getFilterType() { return equalityMatch; };
```

5.81 src/filter_helper_functions.cpp File Reference

```
#include "inc/filter_helper_functions.h"
```

Include dependency graph for filter_helper_functions.cpp:



Functions

- bool substrFilterHandler (SubstringFilter *sf, int *err, std::vector< unsigned char > attribute)

  evaluates if filter is true for given database entry
- bool equalityMatchHandler (EqualityMatchFilter *emf, int *err, std::vector< unsigned char > attribute)

  evalues if filter is true for given database entry
- bool filterLine (FilterObject *f, int *err, DatabaseObject &databaseEntry)

  evaluates if filter is true for given database entry
- std::vector< DatabaseObject > filterHandler (FilterObject *f, int *err, const char *dbLocation, int sizeLimit)

  evaluates if filter is true for given database entries
- FilterObject * convertToFilterObject (std::vector< unsigned char >::iterator BERfilter, std::vector< unsigned char >::iterator end)

  converts BER representation of filters to filter object

5.81.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file filter_helper_functions.cpp.

5.81.2 Function Documentation

5.81.2.1 convertToFilterObject()

```
FilterObject * convertToFilterObject (
            std::vector< unsigned char >::iterator BERfilter,
            std::vector< unsigned char >::iterator end )
```

converts BER representation of filters to filter object

Parameters

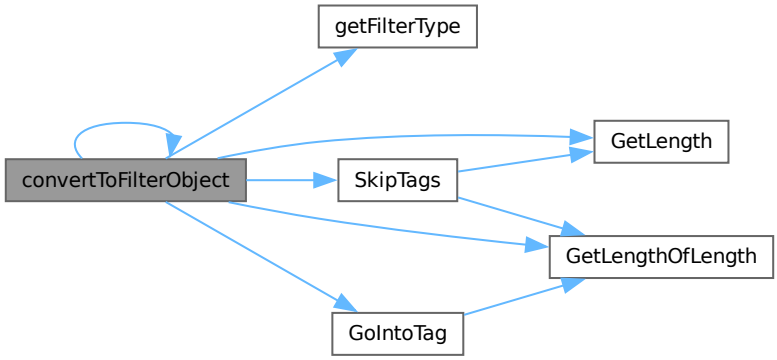
BERfilter	start of the BER filter
end	end of the BER filter

Returns

FilterObject*

Definition at line 204 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.81.2.2 equalityMatchHandler()

```
bool equalityMatchHandler (
            EqualityMatchFilter * emf,
            int * err,
            std::vector< unsigned char > attribute )
```

evalues if filter is true for given database entry

Parameters

emf	equality match filter object
err	1 if error, 0 if success
attribute	attribute to be filtered

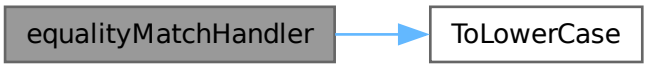
Returns

true

false

Definition at line 83 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.81.2.3 filterHandler()

```
std::vector< DatabaseObject > filterHandler (
            FilterObject * f,
            int * err,
            const char * dbLocation,
            int sizeLimit )
```

evaluates if filter is true for given database entries

Parameters

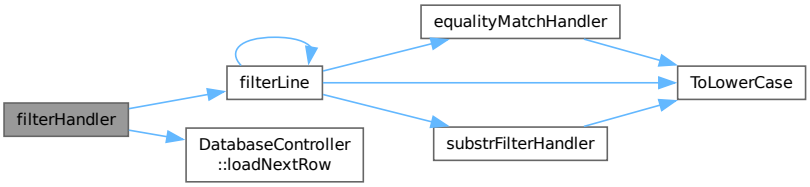
f	filter object
err	1 if error, 0 if success
dbLocation	path to database file
sizeLimit	maximum number of entries to be returned

Returns

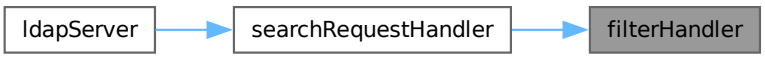
std::vector<DatabaseObject>

Definition at line 175 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.81.2.4 filterLine()

```
bool filterLine (
            FilterObject * f,
            int * err,
            DatabaseObject & databaseEntry )
```

evaluates if filter is true for given database entry

Parameters

f	filter object
err	1 if error, 0 if success
databaseEntry	database entry to be filtered

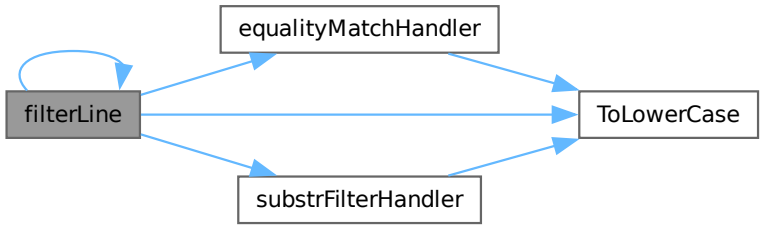
Returns

true

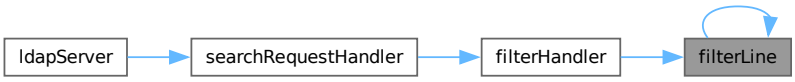
false

Definition at line 91 of file filter_helper_functions.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.81.2.5 substrFilterHandler()

```
bool substrFilterHandler (
            SubstringFilter * sf,
            int * err,
            std::vector< unsigned char > attribute )
```

evaluates if filter is true for given database entry

Parameters

sf	substring filter object
err	1 if error, 0 if success
attribute	attribute to be filtered

Returns

true

false

Definition at line 8 of file filter_helper_functions.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

5.82 filter_helper_functions.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/filter_helper_functions.h"
00007
00008 bool substrFilterHandler(SubstringFilter *sf, int *err,
00009                          std::vector<unsigned char> attribute) {
00010   std::vector<unsigned char> attributeInital = {};
00011   std::vector<unsigned char> attributeMiddle = {};
00012   std::vector<unsigned char> attributeFinal = {};
00013
00014   // extract initial
00015   if (!sf->getSubInitial().empty()) {
00016     if (attribute.size() >= sf->getSubInitial().size()) {
00017       attributeInital = std::vector<unsigned char>(
00018           attribute.begin(), attribute.begin() + sf->getSubInitial().size());
00019       attributeMiddle = std::vector<unsigned char>(
00020           attribute.begin() + sf->getSubInitial().size(), attribute.end());
00021     } else {
00022       return false;
00023     }
00024   } else {
00025     attributeMiddle =
00026         std::vector<unsigned char>(attribute.begin(), attribute.end());
00027   }
00028
00029   // extract final
00030   if (!sf->getSubFinal().empty()) {
00031     if (attribute.size() >= sf->getSubFinal().size()) {
00032       attributeFinal = std::vector<unsigned char>(
00033           attribute.end() - sf->getSubFinal().size(), attribute.end());
00034       if (attributeMiddle.size() > sf->getSubFinal().size()) {
00035         attributeMiddle = std::vector<unsigned char>(
00036             attributeMiddle.begin(),
00037             attributeMiddle.end() - sf->getSubFinal().size());
00038       }
00039     } else {
00040       return false;
00041     }
00042   }
00043
00044   // check subInitial
00045   if (!sf->getSubInitial().empty() &&
00046       ToLowerCase(attributeInital) != ToLowerCase(sf->getSubInitial())) {
00047     return false;
00048   }
00049   // check subFinal
```

```
00050   if (!sf->getSubFinal().empty() &&
00051       ToLowerCase(attributeFinal) != ToLowerCase(sf->getSubFinal())) {
00052     return false;
00053   }
00054   unsigned long int x = 0;
00055
00056   // convert to lower case
00057   attributeMiddle = ToLowerCase(attributeMiddle);
00058
00059   for (unsigned long int y = 0; y < sf->getSubAny().size(); y++) {
00060     bool match = false;
00061     for (; x < attributeMiddle.size(); x++) {
00062
00063       if (attributeMiddle[x] == std::tolower(sf->getSubAny()[y][0])) {
00064         match = true;
00065         for (unsigned long int z = 0; z < sf->getSubAny()[y].size(); z++) {
00066           if (attributeMiddle[x + z] != std::tolower(sf->getSubAny()[y][z])) {
00067             match = false;
00068             break;
00069           }
00070         }
00071         if (match) {
00072           break;
00073         }
00074       }
00075     }
00076     if (!match) {
00077       return false;
00078     }
00079   }
00080   return true;
00081 }
00082
00083 bool equalityMatchHandler(EqualityMatchFilter *emf, int *err,
00084                           std::vector<unsigned char> attribute) {
00085   if (ToLowerCase(emf->getAssertionValue()) == ToLowerCase(attribute)) {
00086     return true;
00087   }
00088   return false;
00089 }
00090
00091 bool filterLine(FilterObject *f, int *err, DatabaseObject &databaseEntry) {
00092
00093   // cn
00094   std::vector<unsigned char> cn = {'c', 'n'};
00095   // CommonName
00096   std::vector<unsigned char> CommonName = {'c', 'o', 'm', 'm', 'o',
00097                                            'n', 'n', 'a', 'm', 'e'};
00098   // email
00099   std::vector<unsigned char> email = {'e', 'm', 'a', 'i', 'l'};
00100   // email
00101   std::vector<unsigned char> mail = {'m', 'a', 'i', 'l'};
00102   // uid
00103   std::vector<unsigned char> uid = {'u', 'i', 'd'};
00104   // UserID
00105   std::vector<unsigned char> UserID = {'u', 's', 'e', 'r', 'i', 'd'};
00106
00107   switch (f->getFilterType()) {
00108   case equalityMatch: {
00109     std::vector<unsigned char> attributeDescription =
00110         ToLowerCase(((EqualityMatchFilter *)f)->getAttributeDescription());
00111     if (attributeDescription == cn || attributeDescription == CommonName) {
00112       return equalityMatchHandler((EqualityMatchFilter *)f, err,
00113                                   databaseEntry.get_name());
00114     } else if (attributeDescription == email || attributeDescription == mail) {
00115       return equalityMatchHandler((EqualityMatchFilter *)f, err,
00116                                   databaseEntry.get_email());
00117     } else if (attributeDescription == uid || attributeDescription == UserID) {
00118       return equalityMatchHandler((EqualityMatchFilter *)f, err,
00119                                   databaseEntry.get_uid());
00120     } else {
00121       *err = 2;
00122       return false;
00123     }
00124
00125   } break;
00126   case substrings: {
00127     std::vector<unsigned char> attributeDescription =
00128           ToLowerCase(((SubstringFilter *)f)->getAttributeDescription());
00129     if (attributeDescription == cn || attributeDescription == CommonName) {
00130       return substrFilterHandler((SubstringFilter *)f, err,
00131                                  databaseEntry.get_name());
00132     } else if (attributeDescription == email || attributeDescription == mail) {
00133       return substrFilterHandler((SubstringFilter *)f, err,
00134                                  databaseEntry.get_email());
00135     } else if (attributeDescription == uid || attributeDescription == UserID) {
00136       return substrFilterHandler((SubstringFilter *)f, err,
```

```
00137                                   databaseEntry.get_uid());
00138     } else {
00139       *err = 2;
00140       return false;
00141     }
00142   } break;
00143   case AND: {
00144     AndFilter *af = (AndFilter *)f;
00145     for (unsigned long int i = 0; i < af->filters.size(); i++) {
00146       if (!filterLine(af->filters[i], err, databaseEntry)) {
00147         return false;
00148       }
00149     }
00150     return true;
00151
00152   } break;
00153   case OR: {
00154     OrFilter *of = (OrFilter *)f;
00155     for (unsigned long int i = 0; i < of->filters.size(); i++) {
00156       if (filterLine(of->filters[i], err, databaseEntry)) {
00157         return true;
00158       }
00159     }
00160     return false;
00161   } break;
00162   case NOT: {
00163     NotFilter *nf = (NotFilter *)f;
00164     bool result = !filterLine((nf->filter), err, databaseEntry);
00165     if (*err == 2)
00166       return false;
00167     return result;
00168   } break;
00169   default:
00170     return false;
00171   }
00172   return false;
00173 }
00174
00175 std::vector<DatabaseObject> filterHandler(FilterObject *f, int *err,
00176                                           const char *dbLocation,
00177                                           int sizeLimit) {
00178   std::vector<DatabaseObject> resultDB;
00179   int dbErr = 0;
00180   DatabaseController db(dbLocation);
00181   int lineCounter = 0;
00182   while (true) {
00183
00184     DatabaseObject obj = db.loadNextRow(&dbErr);
00185
00186     if (dbErr != 0)
00187       break;
00188
00189     if (filterLine(f, err, obj)) {
00190       resultDB.push_back(obj);
00191       lineCounter++;
00192     }
00193
00194     if (lineCounter >= sizeLimit && sizeLimit != 0) {
00195       *err = 1;
00196       break;
00197     }
00198   }
00199
00200   return resultDB;
00201 }
00202
00203 FilterObject *
00204 convertToFilterObject(std::vector<unsigned char>::iterator BERfilter,
00205                       std::vector<unsigned char>::iterator end) {
00206
00207   FilterObject *f;
00208   int err;
00209   int lenght = 0;
00210   int ll = 0;
00211   std::vector<unsigned char> attributeDescription;
00212   std::vector<unsigned char> assertionValue;
00213
00214   switch (getFilterType(BERfilter)) {
00215   case equalityMatch:
00216
00217     GoIntoTag(BERfilter, &err, end);
00218     if (err != 0)
00219       return new FilterObject();
00220     lenght = GetLength(BERfilter + 1, &err, end);
00221
00222     ll = GetLengthOfLength(BERfilter + 1, &err, end);
00223
```

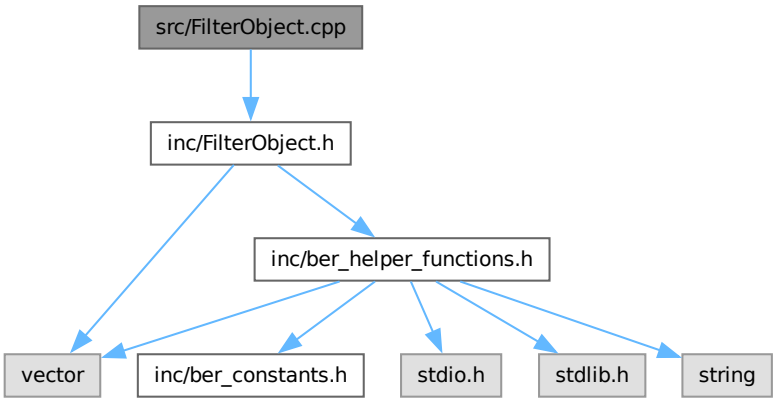
```
00224     for (int i = 0; i < lenght; i++) {
00225       attributeDescription.push_back(BERfilter[1 + ll + i]);
00226     }
00227
00228     SkipTags(BERfilter, 1, &err, end);
00229     if (err != 0)
00230       return new FilterObject();
00231
00232     lenght = GetLength(BERfilter + 1, &err, end);
00233     ll = GetLengthOfLength(BERfilter + 1, &err, end);
00234
00235     for (int i = 0; i < lenght; i++) {
00236       assertionValue.push_back(BERfilter[1 + ll + i]);
00237     }
00238
00239     f = new EqualityMatchFilter(attributeDescription, assertionValue);
00240
00241     break;
00242   case substrings: {
00243     std::vector<unsigned char> attributeDescription;
00244     std::vector<unsigned char> initial;
00245     std::vector<std::vector<unsigned char» any;
00246     std::vector<unsigned char> final;
00247     GoIntoTag(BERfilter, &err, end);
00248     if (err != 0)
00249       return new FilterObject();
00250     lenght = GetLength(BERfilter + 1, &err, end);
00251     ll = GetLengthOfLength(BERfilter + 1, &err, end);
00252
00253     for (int i = 0; i < lenght; i++) {
00254       attributeDescription.push_back(BERfilter[1 + ll + i]);
00255     }
00256
00257     SkipTags(BERfilter, 1, &err, end);
00258     if (err != 0)
00259       return new FilterObject();
00260     int lenghtOfSequence = GetLength(BERfilter + 1, &err, end);
00261     int lenghtOflenOfSequence = GetLengthOfLength(BERfilter + 1, &err, end);
00262     GoIntoTag(BERfilter, &err, end);
00263     if (err != 0)
00264       return new FilterObject();
00265
00266     int currentBitPointer = 0;
00267     while (currentBitPointer < lenghtOfSequence + lenghtOflenOfSequence) {
00268       switch (BERfilter[0]) {
00269       case 0x80: {
00270         int dataLenght = GetLength(BERfilter + 1, &err, end);
00271         int LenghtLenght = GetLengthOfLength(BERfilter + 1, &err, end);
00272         initial = std::vector<unsigned char>(BERfilter + 1 + LenghtLenght,
00273                                              BERfilter + 1 + LenghtLenght +
00274                                                  dataLenght);
00275       } break;
00276       case 0x81: {
00277         int dataLenght = GetLength(BERfilter + 1, &err, end);
00278         int LenghtLenght = GetLengthOfLength(BERfilter + 1, &err, end);
00279         std::vector<unsigned char> tmp = std::vector<unsigned char>(
00280             BERfilter + 1 + LenghtLenght,
00281             BERfilter + 1 + LenghtLenght + dataLenght);
00282         any.push_back(tmp);
00283       } break;
00284       case 0x82: {
00285         int dataLenght = GetLength(BERfilter + 1, &err, end);
00286         int LenghtLenght = GetLengthOfLength(BERfilter + 1, &err, end);
00287         final = std::vector<unsigned char>(BERfilter + 1 + LenghtLenght,
00288                                            BERfilter + 1 + LenghtLenght +
00289                                                dataLenght);
00290       } break;
00291       }
00292       currentBitPointer += 1 + GetLengthOfLength(BERfilter + 1, &err, end) +
00293                            GetLength(BERfilter + 1, &err, end);
00294       SkipTags(BERfilter, 1, &err, end);
00295     }
00296     f = new SubstringFilter(attributeDescription, initial, any, final);
00297
00298   } break;
00299   case AND:
00300     f = new AndFilter();
00301
00302     lenght = GetLength(BERfilter + 1, &err, end);
00303     GoIntoTag(BERfilter, &err, end);
00304     if (err != 0)
00305       return new FilterObject();
00306     for (int i = 0; i < lenght;) {
00307
00308       if (err != 0)
00309         return new FilterObject();
00310       FilterObject *tmpF = convertToFilterObject(BERfilter, end);
```

```
00311       printf("filter type: %d\n", tmpF->getFilterType());
00312       fflush(stdout);
00313       ((AndFilter *)f)->filters.push_back(tmpF);
00314       i += 1 + GetLengthOfLength(BERfilter + 1, &err, end) +
00315            GetLength(BERfilter + 1, &err, end);
00316       SkipTags(BERfilter, 1, &err, end);
00317     }
00318
00319     break;
00320   case OR:
00321     f = new OrFilter();
00322     lenght = GetLength(BERfilter + 1, &err, end);
00323     GoIntoTag(BERfilter, &err, end);
00324     if (err != 0)
00325       return new FilterObject();
00326
00327     for (int i = 0; i < lenght;) {
00328       if (err != 0)
00329         return new FilterObject();
00330       FilterObject *tmpF = convertToFilterObject(BERfilter, end);
00331       ((OrFilter *)f)->filters.push_back(tmpF);
00332       i += 1 + GetLengthOfLength(BERfilter + 1, &err, end) +
00333            GetLength(BERfilter + 1, &err, end);
00334       SkipTags(BERfilter, 1, &err, end);
00335     }
00336
00337     break;
00338   case NOT: {
00339     f = new NotFilter();
00340     lenght = GetLength(BERfilter + 1, &err, end);
00341     GoIntoTag(BERfilter, &err, end);
00342     if (err != 0)
00343       return new FilterObject();
00344     FilterObject *tmpF = convertToFilterObject(BERfilter, end);
00345     ((NotFilter *)f)->filter = tmpF;
00346     break;
00347   }
00348   default:
00349     f = new FilterObject();
00350     break;
00351   }
00352   return f;
00353 }
```

5.83 src/FilterObject.cpp File Reference

`#include "inc/FilterObject.h"`

Include dependency graph for FilterObject.cpp:



5.83.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file FilterObject.cpp.

5.84 FilterObject.cpp

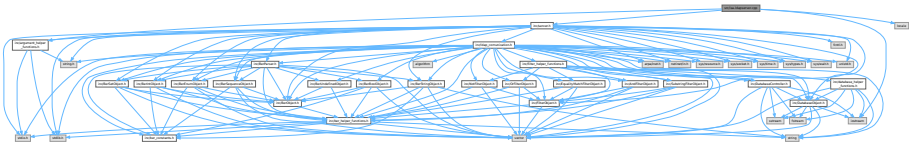
Go to the documentation of this file.

```
00001
00006 #include "inc/FilterObject.h"
00007
00008 filterTypes FilterObject::getFilterType() { return undefined; };
00009
00010 FilterObject::~FilterObject() {}
```

5.85 src/isa-ldapserver.cpp File Reference

```
#include "inc/argument_helper_functions.h"
#include "inc/server.h"
#include <iostream>
#include <locale>
```

Include dependency graph for isa-ldapserver.cpp:



Functions

- bool file_exists (char *name)
- int main (int argc, const char *argv[])

5.85.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file isa-ldapserver.cpp.

5.85.2 Function Documentation

5.85.2.1 file_exists()

```
bool file_exists (
            char * name )
```

Definition at line 11 of file isa-ldapserver.cpp.

5.85.2.2 main()

```
int main (
            int argc,
            const char * argv[] )
```

Definition at line 20 of file isa-ldapserver.cpp.

5.86 isa-ldapserver.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/argument_helper_functions.h"
00007 #include "inc/server.h"
00008 #include <iostream>
00009 #include <locale>
00010
00011 bool file_exists(char *name) {
00012   if (FILE *file = fopen(name, "r")) {
00013     fclose(file);
00014     return true;
00015   } else {
00016     return false;
00017   }
00018 }
00019
00020 int main(int argc, const char *argv[]) {
00021
00022   args_t args = parseArguments(argc, argv);
00023   if (args.err) {
00024     fprintf(stderr, "Error parsing arguments\n");
00025     return 1;
00026   }
00027   if (!file_exists(args.dbPath)) {
00028     std::cout « "File doesn't exist!" « std::endl;
00029     return 1;
00030   }
00031   ldapServer(args.port, args.dbPath);
00032 }
```

5.87 src/ldap_comunication.cpp File Reference

```
#include "inc/ldap_comunication.h"
```

Include dependency graph for ldap_comunication.cpp:

Functions

- BerObject * InitSearchResultEntry (BerObject *searchRequest, std::vector< unsigned char > LDAPDN)
  Initialize the search result entry envelope.
- int AddToSearchResultEntry (BerObject *envelope, std::vector< unsigned char > &attributeDescription, std::vector< unsigned char > &attributeValue)
  Adds an attribute to the search result entry envelope.
- BerObject * CreateBindResponse (BerObject *bindRequest, int resultCode)
  Create a Bind Response object.
- int sendSearchResultDone (BerSequenceObject *searchRequest, int comm_socket, unsigned int result_code)
  sends the search result done envelope to the client
- int checkSearchRequest (BerObject *searchRequest)
  checks if the search request is valid
- int sendNoticeOfDisconnection (int comSocket, char errCode)
  sends notice of disconnection to the client
- int searchRequestHandler (BerObject *searchRequest, int comm_socket, const char *dbPath)
  sends search result entry to the client
- int loadEnvelope (std::vector< unsigned char > &bindRequest, int comm_socket)
  loads the envelope from the client, waits until all the date are received

5.87.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file ldap_comunication.cpp.

5.87.2 Macro Definition Documentation

5.87.2.1 DEBUG

```
#define DEBUG
```

Definition at line 7 of file ldap_comunication.cpp.

5.87.3 Function Documentation

5.87.3.1 AddToSearchResultEntry()

```
int AddToSearchResultEntry (
            BerObject * envelope,
            std::vector< unsigned char > & attributeDescription,
            std::vector< unsigned char > & attributeValue )
```

Adds an attribute to the search result entry envelope.

Parameters

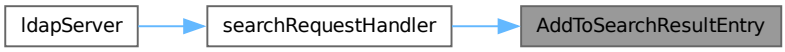
envelope	search result entry envelope
attributeDescription	
attributeValue	

Returns

int

Definition at line 24 of file ldap_comunication.cpp.

Here is the caller graph for this function:



5.87.3.2 checkSearchRequest()

```
int checkSearchRequest (
            BerObject * searchRequest )
```

checks if the search request is valid

Parameters

searchRequest	

Returns

int 0 if valid, -1 if inavalid application sequence, -2 if invalid message id or whole envelope

Definition at line 74 of file ldap_comunication.cpp.

Here is the caller graph for this function:

5.87.3.3 CreateBindResponse()

```
BerObject * CreateBindResponse (
            BerObject * bindRequest,
            int resultCode )
```

Create a Bind Response object.

Parameters

bindRequest	
resultCode	

Returns

BerObject*

Definition at line 43 of file ldap_comunication.cpp.

Here is the caller graph for this function:

5.87.3.4 InitSearchResultEntry()

```
BerObject * InitSearchResultEntry (
            BerObject * searchRequest,
            std::vector< unsigned char > LDAPDN )
```

Initialize the search result entry envelope.

Parameters

searchRequest	search request envelope for which the search result entry
LDAPDN	LDAPDN of the entry

Returns

BerObject*

Definition at line 9 of file ldap_comunication.cpp.

Here is the caller graph for this function:

5.87.3.5 loadEnvelope()

```
int loadEnvelope (
            std::vector< unsigned char > & bindRequest,
            int comm_socket )
```

loads the envelope from the client, waits until all the date are received

Parameters

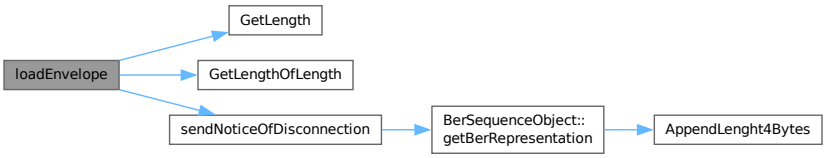
bindRequest	returns the envelope as a vector of unsigned chars
comm_socket	socket to receive the envelope from

Returns

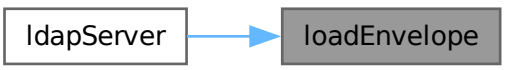
int 0 if success, -1 if error ocured

Definition at line 275 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.87.3.6 searchRequestHandler()

```
int searchRequestHandler (
            BerObject * searchRequest,
            int comm_socket,
            const char * dbPath )
```

sends search result entry to the client

Parameters

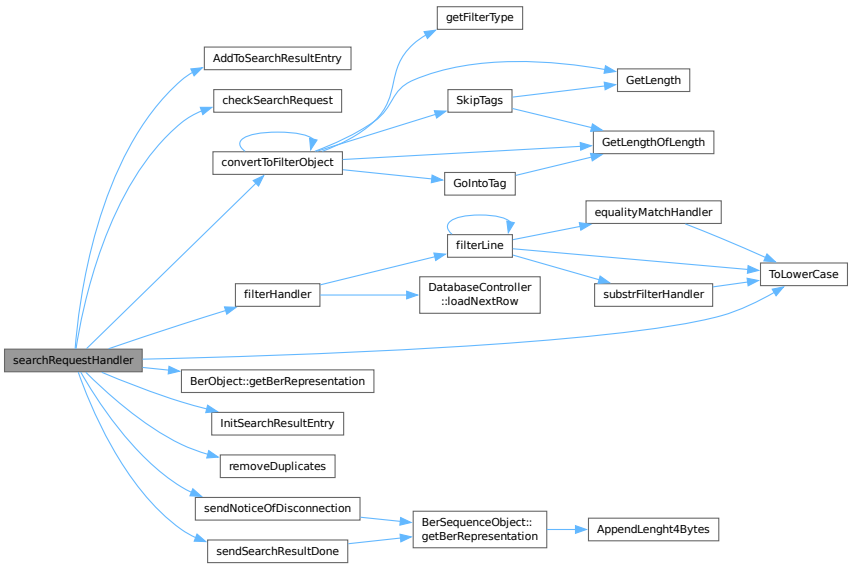
envelope	search request envelope
comSocket	socket to send the envelope to

Returns

int

Definition at line 140 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.87.3.7 sendNoticeOfDisconnection()

```
int sendNoticeOfDisconnection (
            int comSocket,
            char errCode )
```

sends notice of disconnection to the client

Parameters

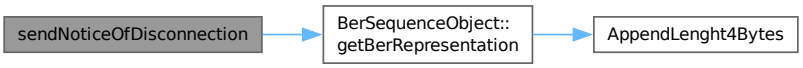
comSocket	socket to send the notice to
errCode	error code

Returns

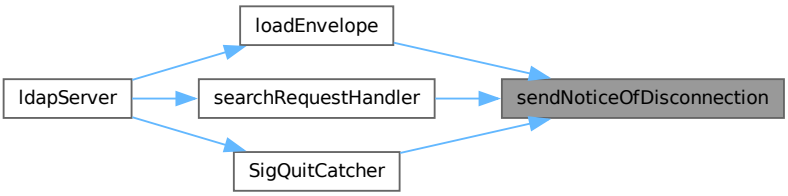
int

Definition at line 125 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.87.3.8 sendSearchResultDone()

```
int sendSearchResultDone (
            BerSequenceObject * searchRequest,
            int comm_socket,
            unsigned int result_code )
```

sends the search result done envelope to the client

Parameters

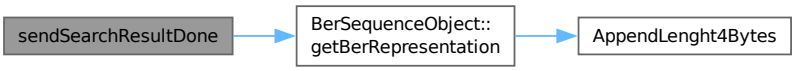
searchRequest	search request envelope for which the search result done is
comm_socket	socket to send the envelope to
result_code	

Returns

int

Definition at line 55 of file ldap_comunication.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.88 ldap_comunication.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/ldap_comunication.h"
00007 #define DEBUG
00008
00009 BerObject *InitSearchResultEntry(BerObject *searchRequest,
00010                                  std::vector<unsigned char> LDAPDN) {
00011   BerSequenceObject *envelope = new BerSequenceObject();
00012   envelope->objects.push_back(new BerIntObject(
00013       static_cast<BerIntObject *>(
00014           static_cast<BerSequenceObject *>(searchRequest)->objects[0])
00015           ->getValue())); // copy message ID
00016   BerSequenceObject *PartialAttributeList =
00017       new BerSequenceObject(BER_SEARCH_RESULT_ENTRY_C);
00018   envelope->objects.push_back(PartialAttributeList);
00019   PartialAttributeList->objects.push_back(new BerStringObject(LDAPDN));
00020   PartialAttributeList->objects.push_back(new BerSequenceObject());
00021   return envelope;
00022 }
00023
00024 int AddToSearchResultEntry(BerObject *envelope,
00025                            std::vector<unsigned char> &attributeDescription,
00026                            std::vector<unsigned char> &attributeValue) {
00027   BerSequenceObject *SearchResultEntry =
00028       (BerSequenceObject *)(((BerSequenceObject *)envelope)->objects[1]);
00029   BerSequenceObject *PartialAttributeList =
```

```
00030       (BerSequenceObject *)(((BerSequenceObject *)SearchResultEntry)
00031                                 ->objects[1]);
00032   BerSequenceObject *attributeValueSequence = new BerSequenceObject();
00033   attributeValueSequence->objects.push_back(
00034       new BerStringObject(attributeDescription));
00035   BerSetObject *attributeValueSet = new BerSetObject();
00036   attributeValueSet->objects.push_back(new BerStringObject(attributeValue));
00037   attributeValueSequence->objects.push_back(attributeValueSet);
00038
00039   PartialAttributeList->objects.push_back(attributeValueSequence);
00040   return 0;
00041 }
00042
00043 BerObject *CreateBindResponse(BerObject *bindRequest, int resultCode) {
00044   BerSequenceObject *envelope = new BerSequenceObject();
00045   envelope->objects.push_back(((BerSequenceObject *)(bindRequest))->objects[0]);
00046   BerSequenceObject *bindResponseSequence =
00047       new BerSequenceObject(BER_BIND_RESPONSE_C);
00048   envelope->objects.push_back(bindResponseSequence);
00049   bindResponseSequence->objects.push_back(new BerEnumObject(resultCode));
00050   bindResponseSequence->objects.push_back(new BerStringObject());
00051   bindResponseSequence->objects.push_back(new BerStringObject());
00052   return envelope;
00053 }
00054
00055 int sendSearchResultDone(BerSequenceObject *searchRequest, int comm_socket,
00056                          unsigned int result_code) {
00057   BerSequenceObject *envelope = new BerSequenceObject();
00058   envelope->objects.push_back(
00059       new BerIntObject(dynamic_cast<BerIntObject *>(searchRequest->objects[0])
00060                            ->getValue())); // copy message ID
00061   BerSequenceObject *searchResultDoneSequence =
00062       new BerSequenceObject(BER_SEARCH_RESULT_DONE_C);
00063   envelope->objects.push_back(searchResultDoneSequence);
00064   searchResultDoneSequence->objects.push_back(new BerEnumObject(result_code));
00065   searchResultDoneSequence->objects.push_back(new BerStringObject(""));
00066   searchResultDoneSequence->objects.push_back(new BerStringObject(""));
00067
00068   std::vector<unsigned char> envelopeBer = envelope->getBerRepresentation();
00069   send(comm_socket, &envelopeBer[0], envelopeBer.size(), MSG_NOSIGNAL);
00070   delete envelope;
00071   return 0;
00072 }
00073
00074 int checkSearchRequest(BerObject *searchRequest) {
00075
00076   // use dynamic casts
00077   BerSequenceObject *envelope =
00078       dynamic_cast<BerSequenceObject *>(searchRequest);
00079   if (envelope == nullptr) {
00080     return -2;
00081   }
00082   // check count of object inside envelope
00083   if (envelope->objects.size() != 2) {
00084     return -1;
00085   }
00086   BerIntObject *messageID = dynamic_cast<BerIntObject *>(envelope->objects[0]);
00087   if (messageID == nullptr) {
00088     return -2;
00089   }
00090
00091   BerSequenceObject *searchRequestSequence =
00092       dynamic_cast<BerSequenceObject *>(envelope->objects[1]);
00093   if (searchRequestSequence == nullptr) {
00094     return -1;
00095   }
00096   // check count of object inside searchRequestSequence
00097   if (searchRequestSequence->objects.size() != 8) {
00098     return -1;
00099   }
00100   BerStringObject *baseObject =
00101       dynamic_cast<BerStringObject *>(searchRequestSequence->objects[0]);
00102   BerEnumObject *scope =
00103       dynamic_cast<BerEnumObject *>(searchRequestSequence->objects[1]);
00104   BerEnumObject *derefAliases =
00105       dynamic_cast<BerEnumObject *>(searchRequestSequence->objects[2]);
00106   BerIntObject *sizeLimit =
00107       dynamic_cast<BerIntObject *>(searchRequestSequence->objects[3]);
00108   BerIntObject *timeLimit =
00109       dynamic_cast<BerIntObject *>(searchRequestSequence->objects[4]);
00110   BerBoolObject *typesOnly =
00111       dynamic_cast<BerBoolObject *>(searchRequestSequence->objects[5]);
00112   BerUndefinedObject *filter =
00113       dynamic_cast<BerUndefinedObject *>(searchRequestSequence->objects[6]);
00114   BerSequenceObject *attributes =
00115       dynamic_cast<BerSequenceObject *>(searchRequestSequence->objects[7]);
00116
```

```
00117   if (messageID == nullptr || baseObject == nullptr || scope == nullptr ||
00118       derefAliases == nullptr || sizeLimit == nullptr || timeLimit == nullptr ||
00119       typesOnly == nullptr || filter == nullptr || attributes == nullptr) {
00120     return -1;
00121   }
00122   return 0;
00123 }
00124
00125 int sendNoticeOfDisconnection(int comSocket, char errCode) {
00126   BerSequenceObject *envelope = new BerSequenceObject();
00127   envelope->objects.push_back(new BerIntObject(0));
00128   BerSequenceObject *extendedResp =
00129       new BerSequenceObject(BER_EXTENDED_RESPONSE_C);
00130   envelope->objects.push_back(extendedResp);
00131   extendedResp->objects.push_back(new BerEnumObject(errCode));
00132   extendedResp->objects.push_back(
00133       new BerStringObject("1.3.6.1.4.1.1466.20036"));
00134   extendedResp->objects.push_back(new BerStringObject(""));
00135   std::vector<unsigned char> envelopeBer = envelope->getBerRepresentation();
00136   send(comSocket, &envelopeBer[0], envelopeBer.size(), MSG_NOSIGNAL);
00137   return 0;
00138 }
00139
00140 int searchRequestHandler(BerObject *searchRequest, int comm_socket,
00141                          const char *dbPath) {
00142
00143   // checks if search request is valid
00144   int err = checkSearchRequest(searchRequest);
00145   BerSequenceObject *envelope =
00146       dynamic_cast<BerSequenceObject *>(searchRequest);
00147   if (err == -2) {
00148     sendNoticeOfDisconnection(comm_socket, BER_LDAP_PROTOCOL_ERROR);
00149     return -1;
00150   }
00151   if (err == -1) {
00152     sendSearchResultDone((BerSequenceObject *)envelope, comm_socket,
00153                          BER_LDAP_SIZE_LIMIT_EXCEEDED);
00154     return -1;
00155   }
00156   // extracting searchRequestSequence from envelope
00157   BerSequenceObject *searchRequestSequence =
00158       (BerSequenceObject *)envelope->objects[1];
00159
00160   // inicialization of searchRequestType
00161   searchRequestType sr;
00162   sr.sizeLimit = 0;
00163   sr.attributes.cn = false;
00164   sr.attributes.email = false;
00165   sr.attributes.uid = false;
00166
00167   // cn
00168   std::vector<unsigned char> cn = {'c', 'n'};
00169   // CommonName
00170   std::vector<unsigned char> CommonName = {'c', 'o', 'm', 'm', 'o',
00171                                            'n', 'n', 'a', 'm', 'e'};
00172   // email
00173   std::vector<unsigned char> email = {'e', 'm', 'a', 'i', 'l'};
00174   // email
00175   std::vector<unsigned char> mail = {'m', 'a', 'i', 'l'};
00176   // uid
00177   std::vector<unsigned char> uid = {'u', 'i', 'd'};
00178   // UserID
00179   std::vector<unsigned char> UserID = {'u', 's', 'e', 'r', 'i', 'd'};
00180
00181   // getting size limit
00182   sr.sizeLimit =
00183       ((BerIntObject *)searchRequestSequence->objects[3])->getValue();
00184   // getting filters and converting them to FilterObject
00185   std::vector<unsigned char> filtersBer =
00186       (searchRequestSequence->objects[6])->getBerRepresentation();
00187   FilterObject *f = convertToFilterObject(filtersBer.begin(), filtersBer.end());
00188
00189   // filtering database
00190   std::vector<DatabaseObject> result;
00191   err = 0;
00192   result = filterHandler(f, &err, dbPath, sr.sizeLimit);
00193
00194   result = removeDuplicates(result);
00195   bool sizeLimitExceeded = false;
00196   if (err == 1) {
00197     sizeLimitExceeded = true;
00198   }
00199
00200   // getting attributes which should be returned
00201   BerSequenceObject *attributesSequence =
00202       (BerSequenceObject *)searchRequestSequence->objects[7];
00203
```

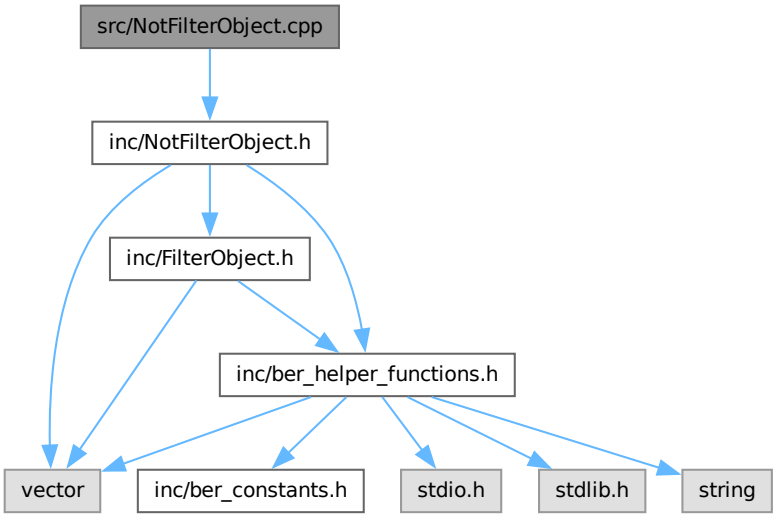
```
00204   for (long unsigned int i = 0; i < attributesSequence->objects.size(); i++) {
00205
00206     if (ToLowerCase(
00207             ((BerStringObject *)attributesSequence->objects[i])->value) == cn ||
00208         ToLowerCase(
00209             ((BerStringObject *)attributesSequence->objects[i])->value) ==
00210             CommonName) {
00211       sr.attributes.cn = true;
00212     }
00213     if (ToLowerCase(
00214             ((BerStringObject *)attributesSequence->objects[i])->value) ==
00215             email ||
00216         ToLowerCase(
00217             ((BerStringObject *)attributesSequence->objects[i])->value) ==
00218             mail) {
00219       sr.attributes.email = true;
00220     }
00221     if (ToLowerCase(
00222             ((BerStringObject *)attributesSequence->objects[i])->value) ==
00223             uid ||
00224         ToLowerCase(
00225             ((BerStringObject *)attributesSequence->objects[i])->value) ==
00226             UserID) {
00227       sr.attributes.uid = true;
00228     }
00229   }
00230   // if no attributes specified, return all
00231   if (!sr.attributes.cn && !sr.attributes.email && !sr.attributes.uid) {
00232     sr.attributes.cn = true;
00233     sr.attributes.email = true;
00234     sr.attributes.uid = true;
00235   }
00236   // send search result entry for each entry in result
00237   for (unsigned long int i = 0; i < result.size(); i++) {
00238     BerObject *searchResultEntry =
00239         InitSearchResultEntry(envelope, result[i].get_uid());
00240
00241     std::vector<unsigned char> resCN = result[i].get_name();
00242     std::vector<unsigned char> resEmail = result[i].get_email();
00243     std::vector<unsigned char> resUID = result[i].get_uid();
00244
00245     if (sr.attributes.cn) {
00246       AddToSearchResultEntry(searchResultEntry, cn, resCN);
00247     }
00248     if (sr.attributes.email) {
00249       AddToSearchResultEntry(searchResultEntry, email, resEmail);
00250     }
00251     if (sr.attributes.uid) {
00252       AddToSearchResultEntry(searchResultEntry, uid, resUID);
00253     }
00254
00255     std::vector<unsigned char> searchResultEntryBer =
00256         searchResultEntry->getBerRepresentation();
00257     send(comm_socket, &searchResultEntryBer[0], searchResultEntryBer.size(),
00258          MSG_NOSIGNAL);
00259     delete searchResultEntry;
00260   }
00261
00262   // send search result done
00263   if (sizeLimitExceeded) {
00264
00265     sendSearchResultDone((BerSequenceObject *)envelope, comm_socket,
00266                          BER_LDAP_SIZE_LIMIT_EXCEEDED);
00267   } else {
00268     sendSearchResultDone((BerSequenceObject *)envelope, comm_socket,
00269                          BER_LDAP_SUCCESS);
00270   }
00271   delete f;
00272   return 0;
00273 }
00274
00275 int loadEnvelope(std::vector<unsigned char> &bindRequest, int comm_socket) {
00276   unsigned char buff[1024];
00277   int lenghtOfMessage = 0;
00278   int err;
00279   int resNow = 0;
00280   int resAll = 0;
00281   for (;;) { // loads lenght of message
00282     int returnCode = recv(comm_socket, buff + resNow, 1024, 0);
00283     if (returnCode == 0)
00284       return -1;
00285     resNow += returnCode;
00286
00287     if (resNow >= 2) {
00288       if ((buff[1] < 0x80) ||
00289           (buff[1] & 0x7F) <= resNow) { // checks if bytes containing lenght of
00290         // message are complete
```

```
00291         bindRequest.insert(bindRequest.end(), buff, buff + resNow);
00292         lenghtOfMessage =
00293             GetLength(bindRequest.begin() + 1, &err, bindRequest.end()) +
00294             GetLengthOfLength(bindRequest.begin() + 1, &err,
00295                               bindRequest.end()) +
00296             1;
00297         break;
00298       }
00299     }
00300   }
00301   resAll = resNow;
00302   for (;;) {
00303     resNow = 0;
00304     if (resAll < lenghtOfMessage) {
00305       int returnCode = recv(comm_socket, buff, 1024, 0);
00306       if (returnCode == 0)
00307         return -1;
00308       resNow += returnCode;
00309       resAll += resNow;
00310     }
00311
00312     if (resAll > 0) {
00313       // check if message is envelope
00314       if (bindRequest[0] != 0x30) {
00315         sendNoticeOfDisconnection(comm_socket, BER_LDAP_PROTOCOL_ERROR);
00316         err = -1;
00317         break;
00318       }
00319       int length = 0;
00320       length = buff[1];
00321
00322       bindRequest.insert(bindRequest.end(), buff, buff + resNow);
00323       // if whole message received, return response
00324       if (resAll >= lenghtOfMessage) {
00325         return length + 2;
00326       }
00327     } else // error or end of connection
00328       break;
00329   }
00330   return -1;
00331 }
```

5.89 src/NotFilterObject.cpp File Reference

`#include "inc/NotFilterObject.h"`

Include dependency graph for NotFilterObject.cpp:



5.89.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file NotFilterObject.cpp.

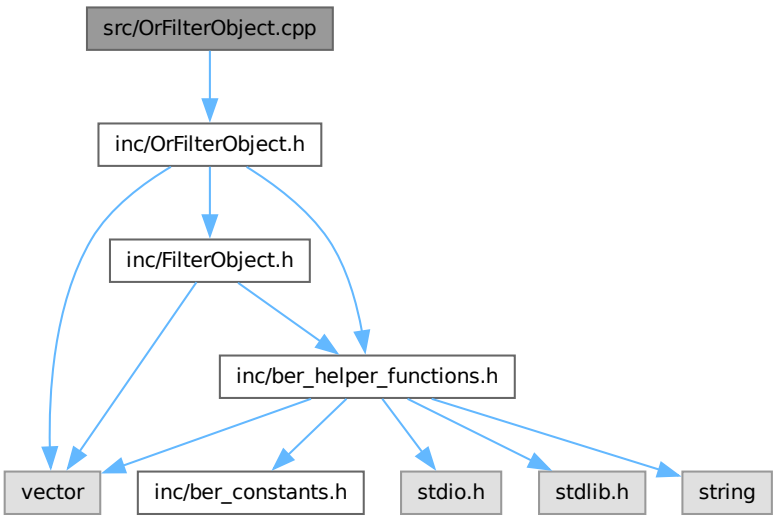
5.90 NotFilterObject.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/NotFilterObject.h"
00007
00008 filterTypes NotFilter::getFilterType() { return NOT; };
00009
00010 NotFilter::~NotFilter() { delete filter; }
```

5.91 src/OrFilterObject.cpp File Reference

`#include "inc/OrFilterObject.h"`
Include dependency graph for OrFilterObject.cpp:



5.91.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file OrFilterObject.cpp.

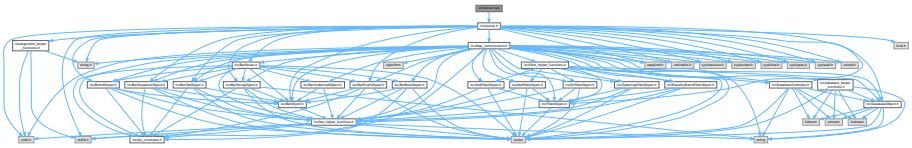
5.92 OrFilterObject.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/OrFilterObject.h"
00007
00008 filterTypes OrFilter::getFilterType() { return OR; };
00009
00010 OrFilter::~OrFilter() {
00011     for (std::vector<FilterObject *>::iterator it = filters.begin();
00012          it != filters.end(); ++it) {
00013         delete *it;
00014     }
00015 }
```

5.93 src/server.cpp File Reference

`#include "inc/server.h"`
Include dependency graph for server.cpp:



Macros

- #define CLEANUP_SERVER delete ((BerSequenceObject *)berBindResponse);

Functions

- void SigCatcher (int n)
- void SigIntCatcher (int n)

  signal handler for main process, when SIGINT is received, it kills all children and closes the socket
- void SigQuitCatcher (int n)

  signal handler for child process, when SIGQUIT is received, it closes the socket and exits
- int ldapServer (int port, char *dbPath)

  Ldap server, This part was inspired by the example from stubs demo tcp server https://git.fit.vutbr.cz/NESFIT/IPK-Projekty/src/branch/master/Stubs/cpp/DemoTcp by Vladimir Vesely Ph.D.

Variables

- int childSocket = 0
- int communicationSocket
- std::vector< int > children = {}

5.93.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file server.cpp.

5.93.2 Macro Definition Documentation

5.93.2.1 CLEANUP_SERVER

```
#define CLEANUP_SERVER delete ((BerSequenceObject *)berBindResponse);
```

Definition at line 48 of file server.cpp.

5.93.3 Function Documentation

5.93.3.1 ldapServer()

```
int ldapServer (
            int port,
            char * dbPath )
```

Ldap server, This part was inspired by the example from stubs demo tcp server https://git.fit.↩vutbr.cz/NESFIT/IPK-Projekty/src/branch/master/Stubs/cpp/DemoTcp by Vladimir Vesely Ph.D.

Parameters

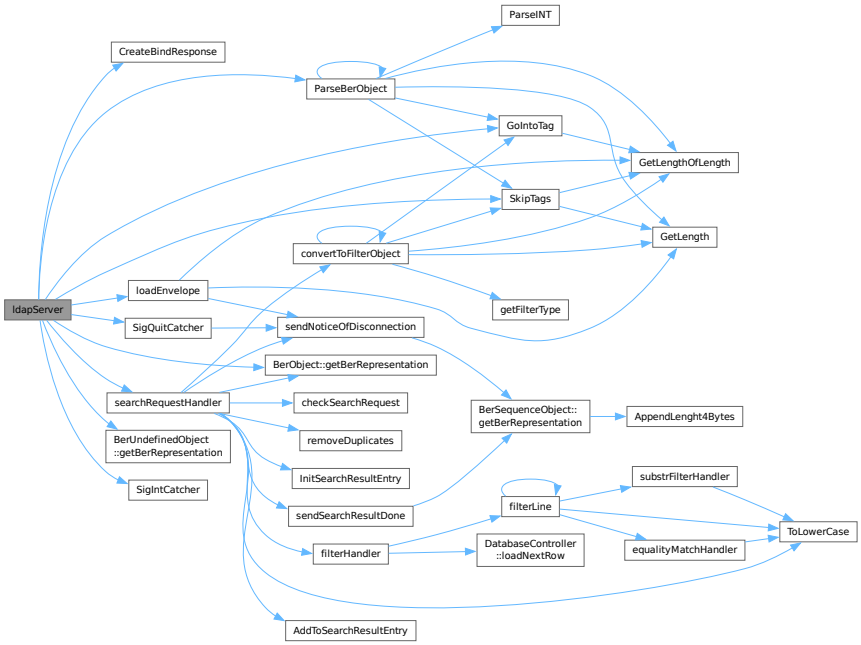
port	port to listen on
dbPath	path to database file

Returns

int

Definition at line 60 of file server.cpp.

Here is the call graph for this function:



5.93.3.2 SigCatcher()

```
void SigCatcher (
            int n )
```

Definition at line 9 of file server.cpp.

5.93.3.3 SigIntCatcher()

```
void SigIntCatcher (
            int n )
```

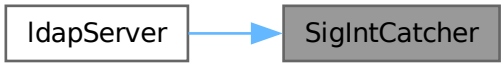
signal handler for main process, when SIGINT is received, it kills all children and closes the socket

Parameters

n	

Definition at line 24 of file server.cpp.

Here is the caller graph for this function:



5.93.3.4 SigQuitCatcher()

```
void SigQuitCatcher (
            int n )
```

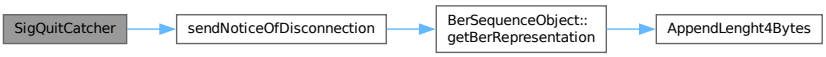
signal handler for child process, when SIGQUIT is received, it closes the socket and exits

Parameters

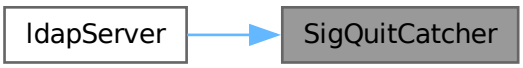
n	

Definition at line 40 of file server.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.93.4 Variable Documentation

5.93.4.1 children

```
std::vector<int> children = {}
```

Definition at line 16 of file server.cpp.

5.93.4.2 childSocket

```
int childSocket = 0
```

Definition at line 14 of file server.cpp.

5.93.4.3 communicationSocket

```
int communicationSocket
```

Definition at line 15 of file server.cpp.

5.94 server.cpp

Go to the documentation of this file.

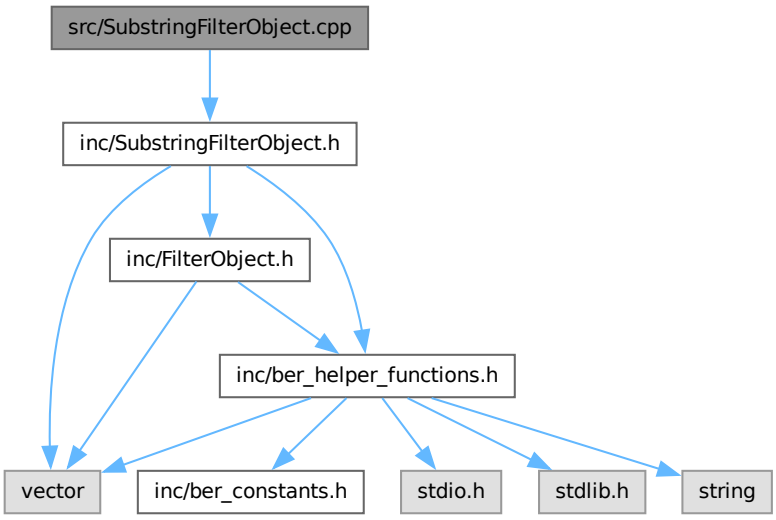
```
00001
00006 #include "inc/server.h"
00007
00008 // Writes which child died
00009 void SigCatcher(int n) {
00010   int pid = wait3(NULL, WNOHANG, NULL);
00011   printf("Child %d died.\n", pid);
00012 }
00013 // global variable for signal handlers
00014 int childSocket = 0;
00015 int communicationSocket;
00016 std::vector<int> children = {};
00017
00024 void SigIntCatcher(int n) {
00025   printf("Killing children.\n");
00026   for (long unsigned int i = 0; i < children.size(); i++) {
00027     kill(children[i], SIGQUIT);
00028   }
00029   close(communicationSocket);
00030   printf("Closing.\n");
00031   exit(0);
00032 }
00033
00040 void SigQuitCatcher(int n) {
00041   sendNoticeOfDisconnection(childSocket, BER_LDAP_UNAVAILABLE);
00042   printf("Dying. %d\n", childSocket);
00043   close(childSocket);
00044   exit(0);
00045 }
00046
00047 // makro for destroying berBindResponse object
00048 #define CLEANUP_SERVER delete ((BerSequenceObject *)berBindResponse);
00049
00060 int ldapServer(int port, char *dbPath) {
00061   int returnCode;
00062   // starting main socket
00063   if ((communicationSocket = socket(PF_INET6, SOCK_STREAM, 0)) < 0) {
00064     perror("socket() failed");
00065     exit(EXIT_FAILURE);
00066   }
00067   // setting options for main socket
00068   const int enable = 1;
00069   const int disable = 0;
00070   returnCode = setsockopt(communicationSocket, IPPROTO_IPV6, IPV6_V6ONLY,
00071                           &disable, sizeof(int));
00072
00073   if (returnCode < 0) {
00074     perror("setsockopt() failed");
00075     close(communicationSocket);
00076     exit(1);
00077   }
00078   returnCode = setsockopt(communicationSocket, SOL_SOCKET, SO_REUSEADDR,
00079                           &enable, sizeof(int));
00080   if (returnCode < 0) {
00081     perror("setsockopt() failed");
00082     close(communicationSocket);
00083     exit(1);
00084   }
```

```
00085   returnCode = setsockopt(communicationSocket, SOL_SOCKET, SO_REUSEPORT,
00086                           &enable, sizeof(int));
00087   if (returnCode < 0) {
00088     perror("setsockopt() failed");
00089     close(communicationSocket);
00090     exit(1);
00091   }
00092
00093   // setting up main socket
00094   struct sockaddr_in6 sa = {0};
00095   struct sockaddr_in6 clientSA = {0};
00096   char str[INET6_ADDRSTRLEN];
00097   socklen_t ClientSALen = sizeof(clientSA);
00098   memset(&sa, 0, sizeof(sa));
00099   sa.sin6_family = AF_INET6;
00100   sa.sin6_addr = in6addr_any;
00101   sa.sin6_port = htons(port);
00102   if ((returnCode =
00103            bind(communicationSocket, (struct sockaddr *)&sa, sizeof(sa))) < 0) {
00104     perror("bind() failed");
00105     exit(EXIT_FAILURE);
00106   }
00107   if ((returnCode = listen(communicationSocket, 1)) < 0) {
00108     perror("listen() failed");
00109     exit(EXIT_FAILURE);
00110   }
00111   // setup signal handlers
00112   //user kills server
00113   signal(SIGINT, SigIntCatcher);
00114   // child dies
00115   signal(SIGCHLD, SigCatcher);
00116   //server kills child
00117   signal(SIGQUIT, SigQuitCatcher);
00118
00119   // main loop
00120   while (1) {
00121     // seting up child socket
00122     childSocket =
00123         accept(communicationSocket, (struct sockaddr *)&clientSA, &ClientSALen);
00124     returnCode = fcntl(communicationSocket, F_SETFD, FD_CLOEXEC);
00125     if (returnCode < 0) {
00126       perror("ERROR: fcntl");
00127       exit(EXIT_FAILURE);
00128     }
00129     if (childSocket <= 0)
00130       continue;
00131
00132     // spliting into child process and parent process
00133     int pid = fork();
00134     if (pid < 0) {
00135       perror("fork() failed");
00136       exit(EXIT_FAILURE);
00137     }
00138
00139     if (pid == 0) // new process to maintain client's requests:
00140     {
00141       int childPid = getpid();
00142       close(communicationSocket); // closing for child process
00143       printf("New ldap connection (maintained by %d):\n", childPid);
00144       if (inet_ntop(AF_INET6, &clientSA.sin6_addr, str, sizeof(str))) {
00145         printf("%d:Client address:%s\n", childPid, str);
00146         printf("%d:Client port: %d\n", childPid, ntohs(clientSA.sin6_port));
00147       }
00148
00149       // load envelope
00150       int err = 0;
00151       std::vector<unsigned char> envelope;
00152       while (1) {
00153         envelope.clear();
00154         int lenghtOfEnvelope = loadEnvelope(envelope, childSocket);
00155         if (lenghtOfEnvelope == -1) {
00156           printf("connection closed\n");
00157           close(childSocket);
00158           exit(0);
00159         }
00160         printf("%d: received message\n", childPid);
00161
00162         std::vector<unsigned char>::iterator envelopeTagPointer =
00163             envelope.begin();
00164
00165         GoIntoTag(envelopeTagPointer, &err, envelope.end());
00166         CHECK_ERR(err, "error going into tag");
00167         SkipTags(envelopeTagPointer, 1, &err, envelope.end());
00168         CHECK_ERR(err, "error skipping tags");
00169         BerObject *EnvelopeObject =
00170             ParseBerObject(envelope.begin(), &err, envelope.end());
00171         switch (envelopeTagPointer[0]) {
```

```
00172         case 0x63:
00173           printf("search request\n");
00174           searchRequestHandler(EnvelopeObject, childSocket, dbPath);
00175           delete EnvelopeObject;
00176           break;
00177         case 0x62: // unbind request
00178           printf("%d: Connection closed\n", childPid);
00179           close(childSocket);
00180           delete EnvelopeObject;
00181           exit(0);
00182           break;
00183         case 0x60: { // bind request
00184           // send bind response
00185           BerObject *berBindResponse =
00186               CreateBindResponse(EnvelopeObject, BER_LDAP_SUCCESS);
00187           std::vector<unsigned char> bindResponse =
00188               berBindResponse->getBerRepresentation();
00189           // check for correct auth method
00190           BerUndefinedObject *authMethod =
00191               (BerUndefinedObject *)((BerSequenceObject *)((BerSequenceObject *)
00192                                                                EnvelopeObject)
00193                                          ->objects[1])
00194                   ->objects[2];
00195           std::vector<unsigned char> authMethodValue =
00196               authMethod->getBerRepresentation();
00197
00198           if (authMethodValue[0] != 0x80 && authMethodValue[1] != 0x00) {
00199             printf("invalid auth method\n");
00200             BerObject *berBindResponse = CreateBindResponse(
00201                 EnvelopeObject, BER_LDAP_AUTH_METHOD_NOT_SUPPORTED);
00202             std::vector<unsigned char> bindResponse =
00203                 berBindResponse->getBerRepresentation();
00204             send(childSocket, &bindResponse[0], bindResponse.size(),
00205                  MSG_NOSIGNAL);
00206             delete EnvelopeObject;
00207             close(childSocket);
00208             exit(0);
00209           }
00210
00211           send(childSocket, &bindResponse[0], bindResponse.size(),
00212                MSG_NOSIGNAL);
00213           delete EnvelopeObject;
00214           break;
00215         }
00216         default:
00217           printf("unknown request\n");
00218           delete EnvelopeObject;
00219           close(childSocket);
00220           exit(0);
00221           break;
00222         }
00223       }
00224
00225       printf("%d: Connection closed\n", childPid);
00226
00227       close(childSocket);
00228       exit(0);
00229     } else {
00230       children.push_back(pid);
00231       close(childSocket);
00232     }
00233   }
00234 }
```

5.95 src/SubstringFilterObject.cpp File Reference

`#include "inc/SubstringFilterObject.h"`
Include dependency graph for SubstringFilterObject.cpp:



5.95.1 Detailed Description

Author

Rene Ceska xceska06 (xceska06@stud.fit.vutbr.cz)

Date

2023-11-19

Definition in file SubstringFilterObject.cpp.

5.96 SubstringFilterObject.cpp

Go to the documentation of this file.

```
00001
00006 #include "inc/SubstringFilterObject.h"
00007
00008 SubstringFilter::SubstringFilter(
00009     std::vector<unsigned char> attributeDescription,
00010     std::vector<unsigned char> subInitial,
00011     std::vector<std::vector<unsigned char» subAny,
00012     std::vector<unsigned char> subFinal) {
00013   this->attributeDescription = attributeDescription;
00014   this->subInitial = subInitial;
00015   this->subAny = subAny;
00016   this->subFinal = subFinal;
00017 };
00018
```

```
00019 std::vector<unsigned char> SubstringFilter::getAttributeDescription() {
00020   return attributeDescription;
00021 };
00022 std::vector<unsigned char> SubstringFilter::getSubInitial() {
00023   return subInitial;
00024 };
00025 std::vector<std::vector<unsigned char» SubstringFilter::getSubAny() {
00026   return subAny;
00027 };
00028 std::vector<unsigned char> SubstringFilter::getSubFinal() { return subFinal; };
00029 filterTypes SubstringFilter::getFilterType() { return substrings; };
```

Index

~AndFilter
  AndFilter, 8
~BerBoolObject
  BerBoolObject, 10
~BerEnumObject
  BerEnumObject, 13
~BerIntObject
  BerIntObject, 15
~BerObject
  BerObject, 18
~BerSequenceObject
  BerSequenceObject, 21
~BerSetObject
  BerSetObject, 24
~DatabaseController
  DatabaseController, 32
~FilterObject
  FilterObject, 37
~NotFilter
  NotFilter, 38
~OrFilter
  OrFilter, 40

AddToSearchResultEntry
  ldap_comunication.cpp, 162
  ldap_comunication.h, 99
AndFilter, 7
  ~AndFilter, 8
  filters, 8
  getFilterType, 8
AppendLenght4Bytes
  ber_helper_functions.cpp, 119
  ber_helper_functions.h, 56
args_t, 8
  dbPath, 9
  err, 9
  port, 9
argument_helper_functions.cpp
  parseArguments, 117
argument_helper_functions.h
  parseArguments, 48
atributeDescriptions
  ldap_comunication.h, 99
attributes
  searchRequest, 42

BER_4BYTE_LENGTH_LENGTH
  ber_constants.h, 50
BER_BIND_REQUEST_C
  ber_constants.h, 50
BER_BIND_RESPONSE_C
  ber_constants.h, 50
BER_BOOL_C
  ber_constants.h, 50
ber_constants.h
  BER_4BYTE_LENGTH_LENGTH, 50
  BER_BIND_REQUEST_C, 50
  BER_BIND_RESPONSE_C, 50
  BER_BOOL_C, 50
  BER_ENUM_C, 50
  BER_EXTENDED_RESPONSE_C, 50
  BER_INT_4BYTES_C, 51
  BER_INT_C, 51
  BER_LDAP_AUTH_METHOD_NOT_SUPPORTED, 51
  BER_LDAP_PROTOCOL_ERROR, 51
  BER_LDAP_SIZE_LIMIT_EXCEEDED, 51
  BER_LDAP_SUCCESS, 51
  BER_LDAP_UNAVAILABLE, 51
  BER_LENGTH_OF_LENGTH_TAG, 51
  BER_OCTET_STRING_C, 52
  BER_SEARCH_REQUEST_C, 52
  BER_SEARCH_RESULT_DONE_C, 52
  BER_SEARCH_RESULT_ENTRY_C, 52
  BER_SEQUENCE_C, 52
  BER_SET_C, 52
  BER_TAG_LENGTH, 52
  BER_UNBIND_REQUEST_C, 52
BER_ENUM_C
  ber_constants.h, 50
BER_EXTENDED_RESPONSE_C
  ber_constants.h, 50
ber_helper_functions.cpp
  AppendLenght4Bytes, 119
  getFilterType, 119
  GetLength, 120
  GetLengthOfLength, 121
  GoIntoTag, 122
  HowManyBytesWillIntUse, 123
  IncreaseLength4Bytes, 123
  ParseINT, 124
  SkipTags, 125
  ToLowerCase, 126
  WriteIntAppend, 127
ber_helper_functions.h
  AppendLenght4Bytes, 56
  berObjectTypes, 55
  filterTypes, 55
  getFilterType, 56

GetLength, 57
GetLengthOfLength, 58
GoIntoTag, 59
HowManyBytesWillIntUse, 60
IncreaseLength4Bytes, 60
ParseINT, 61, 62
SkipTags, 62
ToLowerCase, 63
WriteIntAppend, 64
BER_INT_4BYTES_C
ber_constants.h, 51
BER_INT_C
ber_constants.h, 51
BER_LDAP_AUTH_METHOD_NOT_SUPPORTED
ber_constants.h, 51
BER_LDAP_PROTOCOL_ERROR
ber_constants.h, 51
BER_LDAP_SIZE_LIMIT_EXCEEDED
ber_constants.h, 51
BER_LDAP_SUCCESS
ber_constants.h, 51
BER_LDAP_UNAVAILABLE
ber_constants.h, 51
BER_LENGTH_OF_LENGTH_TAG
ber_constants.h, 51
BER_OCTET_STRING_C
ber_constants.h, 52
BER_SEARCH_REQUEST_C
ber_constants.h, 52
BER_SEARCH_RESULT_DONE_C
ber_constants.h, 52
BER_SEARCH_RESULT_ENTRY_C
ber_constants.h, 52
BER_SEQUENCE_C
ber_constants.h, 52
BER_SET_C
ber_constants.h, 52
BER_TAG_LENGTH
ber_constants.h, 52
BER_UNBIND_REQUEST_C
ber_constants.h, 52
BerBoolObject, 9
~BerBoolObject, 10
BerBoolObject, 10
getBerObjectType, 11
getBerRepresentation, 11
getLenght, 11
BerEnumObject, 12
~BerEnumObject, 13
BerEnumObject, 13
getBerObjectType, 13
getBerRepresentation, 13
getLenght, 13
BerIntObject, 14
~BerIntObject, 15
BerIntObject, 15
getBerObjectType, 15
getBerRepresentation, 15
getLenght, 16
getValue, 16
setValue, 17
BerObject, 17
~BerObject, 18
getBerObjectType, 18
getBerRepresentation, 18
getLenght, 19
berObjectTypes
ber_helper_functions.h, 55
BerParser.cpp
ParseBerObject, 136
BerParser.h
ParseBerObject, 73
BerSequenceObject, 20
~BerSequenceObject, 21
BerSequenceObject, 21
getBerObjectType, 21
getBerRepresentation, 21
getLenght, 22
GetTag, 22
objects, 23
BerSetObject, 23
~BerSetObject, 24
BerSetObject, 24
getBerObjectType, 24
getBerRepresentation, 24
getLenght, 25
objects, 25
BerStringObject, 26
BerStringObject, 27
getBerObjectType, 27
getBerRepresentation, 27
getLenght, 28
value, 28
BerUndefinedObject, 29
BerUndefinedObject, 30
getBerObjectType, 30
getBerRepresentation, 30
getLenght, 30

CHECK_ERR
server.h, 111
checkSearchRequest
ldap_comunication.cpp, 163
ldap_comunication.h, 100
children
server.cpp, 177
childSocket
server.cpp, 177
CLEANUP_SERVER
server.cpp, 175
cn
searchedAttributes, 41
communicationSocket
server.cpp, 178
convertToFilterObject
filter_helper_functions.cpp, 150
filter_helper_functions.h, 90

CreateBindResponse
ldap_comunication.cpp, 163
ldap_comunication.h, 100

database_helper_functions.cpp
removeDuplicates, 145
database_helper_functions.h
removeDuplicates, 83
DatabaseController, 31
~DatabaseController, 32
DatabaseController, 32
loadAllRows, 32
loadNextRow, 32
DatabaseObject, 33
DatabaseObject, 34
get_email, 34
get_name, 34
get_uid, 34
dbPath
args_t, 9
DEBUG
ldap_comunication.cpp, 162

email
searchedAttributes, 41
EqualityMatchFilter, 35
EqualityMatchFilter, 36
getAssertionValue, 36
getAttributeDescription, 36
getFilterType, 36
equalityMatchHandler
filter_helper_functions.cpp, 151
filter_helper_functions.h, 91
err
args_t, 9

file_exists
isa-ldapserver.cpp, 161
filter
NotFilter, 39
filter_helper_functions.cpp
convertToFilterObject, 150
equalityMatchHandler, 151
filterHandler, 152
filterLine, 153
substrFilterHandler, 154
filter_helper_functions.h
convertToFilterObject, 90
equalityMatchHandler, 91
filterHandler, 92
filterLine, 93
substrFilterHandler, 94
filterHandler
filter_helper_functions.cpp, 152
filter_helper_functions.h, 92
filterLine
filter_helper_functions.cpp, 153
filter_helper_functions.h, 93
FilterObject, 36
~FilterObject, 37
getFilterType, 37
filters
AndFilter, 8
OrFilter, 40
filterTypes
ber_helper_functions.h, 55

get_email
DatabaseObject, 34
get_name
DatabaseObject, 34
get_uid
DatabaseObject, 34
getAssertionValue
EqualityMatchFilter, 36
getAttributeDescription
EqualityMatchFilter, 36
SubstringFilter, 44
getBerObjectType
BerBoolObject, 11
BerEnumObject, 13
BerIntObject, 15
BerObject, 18
BerSequenceObject, 21
BerSetObject, 24
BerStringObject, 27
BerUndefinedObject, 30
getBerRepresentation
BerBoolObject, 11
BerEnumObject, 13
BerIntObject, 15
BerObject, 18
BerSequenceObject, 21
BerSetObject, 24
BerStringObject, 27
BerUndefinedObject, 30
getFilterType
AndFilter, 8
ber_helper_functions.cpp, 119
ber_helper_functions.h, 56
EqualityMatchFilter, 36
FilterObject, 37
NotFilter, 38
OrFilter, 40
SubstringFilter, 44
getLenght
BerBoolObject, 11
BerEnumObject, 13
BerIntObject, 16
BerObject, 19
BerSequenceObject, 22
BerSetObject, 25
BerStringObject, 28
BerUndefinedObject, 30
GetLength
ber_helper_functions.cpp, 120
ber_helper_functions.h, 57
GetLengthOfLength

ber_helper_functions.cpp, 121
ber_helper_functions.h, 58
getSubAny
SubstringFilter, 44
getSubFinal
SubstringFilter, 44
getSubInitial
SubstringFilter, 44
GetTag
BerSequenceObject, 22
getValue
BerIntObject, 16
GoIntoTag
ber_helper_functions.cpp, 122
ber_helper_functions.h, 59

HowManyBytesWillIntUse
ber_helper_functions.cpp, 123
ber_helper_functions.h, 60

inc/AndFilterObject.h, 45, 46
inc/argument_helper_functions.h, 47, 48
inc/ber_constants.h, 49, 53
inc/ber_helper_functions.h, 53, 65
inc/BerBoolObject.h, 65, 67
inc/BerEnumObject.h, 67, 68
inc/BerIntObject.h, 69, 70
inc/BerObject.h, 70, 71
inc/BerParser.h, 72, 74
inc/BerSequenceObject.h, 75, 76
inc/BerSetObject.h, 76, 78
inc/BerStringObject.h, 78, 79
inc/BerUndefinedObject.h, 80, 81
inc/database_helper_functions.h, 81, 83
inc/DatabaseController.h, 84, 85
inc/DatabaseObject.h, 85, 87
inc/EqualityMatchFilterObject.h, 87, 88
inc/filter_helper_functions.h, 89, 95
inc/FilterObject.h, 95, 97
inc/ldap_comunication.h, 97, 105
inc/NotFilterObject.h, 106, 108
inc/OrFilterObject.h, 108, 109
inc/server.h, 110, 112
inc/SubstringFilterObject.h, 113, 114
IncreaseLength4Bytes
ber_helper_functions.cpp, 123
ber_helper_functions.h, 60
InitSearchResultEntry
ldap_comunication.cpp, 164
ldap_comunication.h, 101
isa-ldapserver.cpp
file_exists, 161
main, 161

ldap_comunication.cpp
AddToSearchResultEntry, 162
checkSearchRequest, 163
CreateBindResponse, 163
DEBUG, 162
InitSearchResultEntry, 164
loadEnvelope, 165
searchRequestHandler, 165
sendNoticeOfDisconnection, 166
sendSearchResultDone, 167
ldap_comunication.h
AddToSearchResultEntry, 99
atributeDescriptions, 99
checkSearchRequest, 100
CreateBindResponse, 100
InitSearchResultEntry, 101
loadEnvelope, 101
searchRequestHandler, 102
sendNoticeOfDisconnection, 103
sendSearchResultDone, 104
ldapServer
server.cpp, 175
server.h, 111
loadAllRows
DatabaseController, 32
loadEnvelope
ldap_comunication.cpp, 165
ldap_comunication.h, 101
loadNextRow
DatabaseController, 32

main
isa-ldapserver.cpp, 161
messageIDLength
searchRequest, 42

NotFilter, 37
∼NotFilter, 38
filter, 39
getFilterType, 38

objects
BerSequenceObject, 23
BerSetObject, 25
OrFilter, 39
∼OrFilter, 40
filters, 40
getFilterType, 40

parseArguments
argument_helper_functions.cpp, 117
argument_helper_functions.h, 48
ParseBerObject
BerParser.cpp, 136
BerParser.h, 73
ParseINT
ber_helper_functions.cpp, 124
ber_helper_functions.h, 61, 62
port
args_t, 9

removeDuplicates
database_helper_functions.cpp, 145
database_helper_functions.h, 83

searchedAttributes, 40
cn, 41
email, 41
uid, 41
searchRequest, 41
attributes, 42
messageIDLength, 42
sizeLimit, 42
searchRequestHandler
ldap_comunication.cpp, 165
ldap_comunication.h, 102
sendNoticeOfDisconnection
ldap_comunication.cpp, 166
ldap_comunication.h, 103
sendSearchResultDone
ldap_comunication.cpp, 167
ldap_comunication.h, 104
server.cpp
children, 177
childSocket, 177
CLEANUP_SERVER, 175
communicationSocket, 178
ldapServer, 175
SigCatcher, 176
SigIntCatcher, 176
SigQuitCatcher, 177
server.h
CHECK_ERR, 111
ldapServer, 111
setValue
BerIntObject, 17
SigCatcher
server.cpp, 176
SigIntCatcher
server.cpp, 176
SigQuitCatcher
server.cpp, 177
sizeLimit
searchRequest, 42
SkipTags
ber_helper_functions.cpp, 125
ber_helper_functions.h, 62
src/AndFilterObject.cpp, 115, 116
src/argument_helper_functions.cpp, 116, 117
src/ber_helper_functions.cpp, 118, 127
src/BerBoolObject.cpp, 131
src/BerEnumObject.cpp, 132, 133
src/BerIntObject.cpp, 133, 134
src/BerObject.cpp, 134, 135
src/BerParser.cpp, 135, 137
src/BerSequenceObject.cpp, 139, 140
src/BerSetObject.cpp, 141
src/BerStringObject.cpp, 142, 143
src/BerUndefinedObject.cpp, 143, 144
src/database_helper_functions.cpp, 144, 146
src/DatabaseController.cpp, 146, 147
src/DatabaseObject.cpp, 148
src/EqualityMatchFilterObject.cpp, 149
src/filter_helper_functions.cpp, 150, 155
src/FilterObject.cpp, 159, 160
src/isa-ldapserver.cpp, 160, 161
src/ldap_comunication.cpp, 161, 168
src/NotFilterObject.cpp, 172, 173
src/OrFilterObject.cpp, 173, 174
src/server.cpp, 174, 178
src/SubstringFilterObject.cpp, 181
substrFilterHandler
filter_helper_functions.cpp, 154
filter_helper_functions.h, 94
SubstringFilter, 43
getAttributeDescription, 44
getFilterType, 44
getSubAny, 44
getSubFinal, 44
getSubInitial, 44
SubstringFilter, 44

ToLowerCase
ber_helper_functions.cpp, 126
ber_helper_functions.h, 63

uid
searchedAttributes, 41

value
BerStringObject, 28

WriteIntAppend
ber_helper_functions.cpp, 127
ber_helper_functions.h, 64