

xapian-core  
1.0.23

Generated by Doxygen 1.5.9

Fri Jan 14 04:49:14 2011



# Contents

<b>1</b>	<b>Deprecated List</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>11</b>
5.1	File List . . . . .	11
<b>6</b>	<b>Namespace Documentation</b>	<b>13</b>
6.1	Xpian Namespace Reference . . . . .	13
6.1.1	Detailed Description . . . . .	20
6.1.2	Typedef Documentation . . . . .	20
6.1.2.1	doccount . . . . .	20
6.1.2.2	doccount_diff . . . . .	20
6.1.2.3	docid . . . . .	20
6.1.2.4	doclength . . . . .	21
6.1.2.5	percent . . . . .	21
6.1.2.6	termcount . . . . .	21
6.1.2.7	termcount_diff . . . . .	21
6.1.2.8	termpos . . . . .	21
6.1.2.9	termpos_diff . . . . .	21

6.1.2.10	timeout	21
6.1.2.11	valueno	21
6.1.2.12	valueno_diff	22
6.1.2.13	weight	22
6.1.3	Function Documentation	22
6.1.3.1	major_version	22
6.1.3.2	minor_version	22
6.1.3.3	operator!=	22
6.1.3.4	operator!=	22
6.1.3.5	operator!=	22
6.1.3.6	operator!=	22
6.1.3.7	operator!=	23
6.1.3.8	operator!=	23
6.1.3.9	operator==	23
6.1.3.10	operator==	23
6.1.3.11	operator==	23
6.1.3.12	operator==	23
6.1.3.13	operator==	23
6.1.3.14	operator==	23
6.1.3.15	revision	23
6.1.3.16	sortable_serialise	24
6.1.3.17	sortable_unserialise	24
6.1.3.18	version_string	24
6.1.3.19	xapian_major_version	24
6.1.3.20	xapian_minor_version	25
6.1.3.21	xapian_revision	25
6.1.3.22	xapian_version_string	25
6.1.4	Variable Documentation	25
6.1.4.1	BAD_VALUENO	25
6.1.4.2	DB_CREATE	25
6.1.4.3	DB_CREATE_OR_OPEN	25
6.1.4.4	DB_CREATE_OR_OVERWRITE	25
6.1.4.5	DB_OPEN	25
6.2	Xapian::Auto Namespace Reference	26

6.2.1	Detailed Description	26
6.2.2	Function Documentation	26
6.2.2.1	open_stub	26
6.3	Xapian::Flint Namespace Reference	27
6.3.1	Detailed Description	27
6.3.2	Function Documentation	27
6.3.2.1	open	27
6.3.2.2	open	27
6.4	Xapian::InMemory Namespace Reference	29
6.4.1	Detailed Description	29
6.4.2	Function Documentation	29
6.4.2.1	open	29
6.5	Xapian::Quartz Namespace Reference	30
6.5.1	Detailed Description	30
6.5.2	Function Documentation	30
6.5.2.1	open	30
6.5.2.2	WritableDatabaseopen	30
6.6	Xapian::Remote Namespace Reference	32
6.6.1	Detailed Description	32
6.6.2	Function Documentation	32
6.6.2.1	open	32
6.6.2.2	open	33
6.6.2.3	open_writable	33
6.6.2.4	open_writable	34
6.7	Xapian::Unicode Namespace Reference	35
6.7.1	Detailed Description	36
6.7.2	Enumeration Type Documentation	36
6.7.2.1	category	36
6.7.3	Function Documentation	36
6.7.3.1	append_utf8	36
6.7.3.2	get_category	36
6.7.3.3	is_currency	37
6.7.3.4	is_whitespace	37
6.7.3.5	is_wordchar	37

6.7.3.6	<a href="#">nonascii_to_utf8</a>	37
6.7.3.7	<a href="#">to_utf8</a>	37
6.7.3.8	<a href="#">tolower</a>	37
6.7.3.9	<a href="#">tolower</a>	38
6.7.3.10	<a href="#">toupper</a>	38
6.7.3.11	<a href="#">toupper</a>	38
<b>7</b>	<b>Class Documentation</b>	<b>39</b>
7.1	<a href="#">Xapian::AssertionError Class Reference</a>	39
7.1.1	<a href="#">Detailed Description</a>	40
7.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	40
7.1.2.1	<a href="#">AssertionError</a>	40
7.1.2.2	<a href="#">AssertionError</a>	40
7.2	<a href="#">Xapian::BM25Weight Class Reference</a>	41
7.2.1	<a href="#">Detailed Description</a>	42
7.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	42
7.2.2.1	<a href="#">BM25Weight</a>	42
7.2.3	<a href="#">Member Function Documentation</a>	43
7.2.3.1	<a href="#">clone</a>	43
7.2.3.2	<a href="#">get_maxextra</a>	43
7.2.3.3	<a href="#">get_maxpart</a>	43
7.2.3.4	<a href="#">get_sumextra</a>	43
7.2.3.5	<a href="#">get_sumpart</a>	43
7.2.3.6	<a href="#">get_sumpart_needs_doclength</a>	44
7.2.3.7	<a href="#">name</a>	44
7.2.3.8	<a href="#">serialise</a>	44
7.2.3.9	<a href="#">unserialise</a>	44
7.3	<a href="#">Xapian::BoolWeight Class Reference</a>	45
7.3.1	<a href="#">Detailed Description</a>	46
7.3.2	<a href="#">Member Function Documentation</a>	46
7.3.2.1	<a href="#">clone</a>	46
7.3.2.2	<a href="#">get_maxextra</a>	46
7.3.2.3	<a href="#">get_maxpart</a>	46
7.3.2.4	<a href="#">get_sumextra</a>	46

7.3.2.5	<a href="#">get_sumpart</a>	47
7.3.2.6	<a href="#">get_sumpart_needs_doclength</a>	47
7.3.2.7	<a href="#">name</a>	47
7.3.2.8	<a href="#">serialise</a>	47
7.3.2.9	<a href="#">unserialise</a>	47
7.4	<a href="#">Xapian::Database Class Reference</a>	48
7.4.1	<a href="#">Detailed Description</a>	51
7.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	51
7.4.2.1	<a href="#">Database</a>	51
7.4.2.2	<a href="#">Database</a>	51
7.4.2.3	<a href="#">~Database</a>	51
7.4.2.4	<a href="#">Database</a>	51
7.4.3	<a href="#">Member Function Documentation</a>	51
7.4.3.1	<a href="#">add_database</a>	51
7.4.3.2	<a href="#">allterms_begin</a>	52
7.4.3.3	<a href="#">allterms_begin</a>	52
7.4.3.4	<a href="#">allterms_end</a>	52
7.4.3.5	<a href="#">allterms_end</a>	52
7.4.3.6	<a href="#">get_avlength</a>	52
7.4.3.7	<a href="#">get_collection_freq</a>	52
7.4.3.8	<a href="#">get_description</a>	53
7.4.3.9	<a href="#">get_doccount</a>	53
7.4.3.10	<a href="#">get_doclength</a>	53
7.4.3.11	<a href="#">get_document</a>	53
7.4.3.12	<a href="#">get_lastdocid</a>	53
7.4.3.13	<a href="#">get_metadata</a>	53
7.4.3.14	<a href="#">get_spelling_suggestion</a>	54
7.4.3.15	<a href="#">get_termfreq</a>	54
7.4.3.16	<a href="#">has_positions</a>	54
7.4.3.17	<a href="#">keep_alive</a>	54
7.4.3.18	<a href="#">metadata_keys_begin</a>	55
7.4.3.19	<a href="#">metadata_keys_end</a>	55
7.4.3.20	<a href="#">operator=</a>	55
7.4.3.21	<a href="#">positionlist_begin</a>	55

7.4.3.22	<a href="#">positionlist_end</a>	55
7.4.3.23	<a href="#">postlist_begin</a>	55
7.4.3.24	<a href="#">postlist_end</a>	56
7.4.3.25	<a href="#">reopen</a>	56
7.4.3.26	<a href="#">spellings_begin</a>	56
7.4.3.27	<a href="#">spellings_end</a>	56
7.4.3.28	<a href="#">synonym_keys_begin</a>	56
7.4.3.29	<a href="#">synonym_keys_end</a>	56
7.4.3.30	<a href="#">synonyms_begin</a>	57
7.4.3.31	<a href="#">synonyms_end</a>	57
7.4.3.32	<a href="#">term_exists</a>	57
7.4.3.33	<a href="#">termlist_begin</a>	57
7.4.3.34	<a href="#">termlist_end</a>	57
7.5	<a href="#">Xapian::DatabaseCorruptError Class Reference</a>	58
7.5.1	<a href="#">Detailed Description</a>	58
7.5.2	<a href="#">Constructor &amp; Destructor Documentation</a>	59
7.5.2.1	<a href="#">DatabaseCorruptError</a>	59
7.5.2.2	<a href="#">DatabaseCorruptError</a>	59
7.6	<a href="#">Xapian::DatabaseCreateError Class Reference</a>	60
7.6.1	<a href="#">Detailed Description</a>	60
7.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	61
7.6.2.1	<a href="#">DatabaseCreateError</a>	61
7.6.2.2	<a href="#">DatabaseCreateError</a>	61
7.7	<a href="#">Xapian::DatabaseError Class Reference</a>	62
7.7.1	<a href="#">Detailed Description</a>	62
7.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	62
7.7.2.1	<a href="#">DatabaseError</a>	62
7.7.2.2	<a href="#">DatabaseError</a>	62
7.8	<a href="#">Xapian::DatabaseLockError Class Reference</a>	63
7.8.1	<a href="#">Detailed Description</a>	63
7.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	64
7.8.2.1	<a href="#">DatabaseLockError</a>	64
7.8.2.2	<a href="#">DatabaseLockError</a>	64
7.9	<a href="#">Xapian::DatabaseModifiedError Class Reference</a>	65



7.9.1	Detailed Description	65
7.9.2	Constructor & Destructor Documentation	66
7.9.2.1	DatabaseModifiedError	66
7.9.2.2	DatabaseModifiedError	66
7.10	Xapian::DatabaseOpeningError Class Reference	67
7.10.1	Detailed Description	67
7.10.2	Constructor & Destructor Documentation	68
7.10.2.1	DatabaseOpeningError	68
7.10.2.2	DatabaseOpeningError	68
7.11	Xapian::DatabaseVersionError Class Reference	69
7.11.1	Detailed Description	69
7.11.2	Constructor & Destructor Documentation	70
7.11.2.1	DatabaseVersionError	70
7.11.2.2	DatabaseVersionError	70
7.12	Xapian::DateValueRangeProcessor Class Reference	71
7.12.1	Detailed Description	71
7.12.2	Constructor & Destructor Documentation	71
7.12.2.1	DateValueRangeProcessor	71
7.12.3	Member Function Documentation	72
7.12.3.1	operator()	72
7.13	Xapian::DocNotFoundError Class Reference	73
7.13.1	Detailed Description	73
7.13.2	Constructor & Destructor Documentation	73
7.13.2.1	DocNotFoundError	73
7.13.2.2	DocNotFoundError	74
7.14	Xapian::Document Class Reference	75
7.14.1	Detailed Description	76
7.14.2	Constructor & Destructor Documentation	77
7.14.2.1	Document	77
7.14.2.2	Document	77
7.14.2.3	~Document	77
7.14.3	Member Function Documentation	77
7.14.3.1	add_boolean_term	77
7.14.3.2	add_posting	77

7.14.3.3	<a href="#">add_term</a>	78
7.14.3.4	<a href="#">add_value</a>	78
7.14.3.5	<a href="#">clear_terms</a>	78
7.14.3.6	<a href="#">clear_values</a>	78
7.14.3.7	<a href="#">get_data</a>	78
7.14.3.8	<a href="#">get_description</a>	78
7.14.3.9	<a href="#">get_docid</a>	79
7.14.3.10	<a href="#">get_value</a>	79
7.14.3.11	<a href="#">operator=</a>	79
7.14.3.12	<a href="#">remove_posting</a>	79
7.14.3.13	<a href="#">remove_term</a>	80
7.14.3.14	<a href="#">remove_value</a>	80
7.14.3.15	<a href="#">set_data</a>	80
7.14.3.16	<a href="#">termlist_begin</a>	80
7.14.3.17	<a href="#">termlist_count</a>	80
7.14.3.18	<a href="#">termlist_end</a>	80
7.14.3.19	<a href="#">values_begin</a>	80
7.14.3.20	<a href="#">values_count</a>	80
7.14.3.21	<a href="#">values_end</a>	81
7.15	<a href="#">Xapian::Enquire Class Reference</a>	82
7.15.1	<a href="#">Detailed Description</a>	84
7.15.2	<a href="#">Constructor &amp; Destructor Documentation</a>	85
7.15.2.1	<a href="#">Enquire</a>	85
7.15.2.2	<a href="#">Enquire</a>	85
7.15.2.3	<a href="#">~Enquire</a>	85
7.15.3	<a href="#">Member Function Documentation</a>	85
7.15.3.1	<a href="#">get_description</a>	85
7.15.3.2	<a href="#">get_eset</a>	85
7.15.3.3	<a href="#">get_eset</a>	86
7.15.3.4	<a href="#">get_matching_terms_begin</a>	86
7.15.3.5	<a href="#">get_matching_terms_begin</a>	87
7.15.3.6	<a href="#">get_matching_terms_end</a>	88
7.15.3.7	<a href="#">get_matching_terms_end</a>	88
7.15.3.8	<a href="#">get_mset</a>	88

7.15.3.9	<a href="#">get_query</a>	89
7.15.3.10	<a href="#">operator=</a>	89
7.15.3.11	<a href="#">register_match_decider</a>	89
7.15.3.12	<a href="#">set_collapse_key</a>	89
7.15.3.13	<a href="#">set_cutoff</a>	90
7.15.3.14	<a href="#">set_docid_order</a>	90
7.15.3.15	<a href="#">set_query</a>	91
7.15.3.16	<a href="#">set_sort_by_key</a>	91
7.15.3.17	<a href="#">set_sort_by_key_then_relevance</a>	91
7.15.3.18	<a href="#">set_sort_by_relevance</a>	92
7.15.3.19	<a href="#">set_sort_by_relevance_then_key</a>	92
7.15.3.20	<a href="#">set_sort_by_relevance_then_value</a>	92
7.15.3.21	<a href="#">set_sort_by_value</a>	92
7.15.3.22	<a href="#">set_sort_by_value_then_relevance</a>	93
7.15.3.23	<a href="#">set_weighting_scheme</a>	93
7.15.4	<a href="#">Member Data Documentation</a>	93
7.15.4.1	<a href="#">include_query_terms</a>	93
7.15.4.2	<a href="#">use_exact_termfreq</a>	94
7.16	<a href="#">Xapian::Error Class Reference</a>	95
7.16.1	<a href="#">Detailed Description</a>	96
7.16.2	<a href="#">Member Function Documentation</a>	96
7.16.2.1	<a href="#">get_context</a>	96
7.16.2.2	<a href="#">get_description</a>	96
7.16.2.3	<a href="#">get_errno</a>	96
7.16.2.4	<a href="#">get_error_string</a>	96
7.16.2.5	<a href="#">get_msg</a>	96
7.16.2.6	<a href="#">get_type</a>	97
7.17	<a href="#">Xapian::ErrorHandler Class Reference</a>	98
7.17.1	<a href="#">Detailed Description</a>	98
7.17.2	<a href="#">Constructor &amp; Destructor Documentation</a>	98
7.17.2.1	<a href="#">ErrorHandler</a>	98
7.17.2.2	<a href="#">~ErrorHandler</a>	98
7.17.3	<a href="#">Member Function Documentation</a>	98
7.17.3.1	<a href="#">operator()</a>	98

7.18 Xapian::ESet Class Reference . . . . .	100
7.18.1 Detailed Description . . . . .	101
7.18.2 Constructor & Destructor Documentation . . . . .	101
7.18.2.1 ESet . . . . .	101
7.18.2.2 ~ESet . . . . .	101
7.18.2.3 ESet . . . . .	101
7.18.3 Member Function Documentation . . . . .	101
7.18.3.1 back . . . . .	101
7.18.3.2 begin . . . . .	101
7.18.3.3 empty . . . . .	101
7.18.3.4 end . . . . .	102
7.18.3.5 get_description . . . . .	102
7.18.3.6 get_ebound . . . . .	102
7.18.3.7 max_size . . . . .	102
7.18.3.8 operator= . . . . .	102
7.18.3.9 operator[] . . . . .	102
7.18.3.10 size . . . . .	102
7.18.3.11 swap . . . . .	102
7.19 Xapian::ESetIterator Class Reference . . . . .	103
7.19.1 Detailed Description . . . . .	104
7.19.2 Member Typedef Documentation . . . . .	104
7.19.2.1 difference_type . . . . .	104
7.19.2.2 iterator_category . . . . .	104
7.19.2.3 pointer . . . . .	104
7.19.2.4 reference . . . . .	104
7.19.2.5 value_type . . . . .	104
7.19.3 Constructor & Destructor Documentation . . . . .	105
7.19.3.1 ESetIterator . . . . .	105
7.19.3.2 ESetIterator . . . . .	105
7.19.4 Member Function Documentation . . . . .	105
7.19.4.1 get_description . . . . .	105
7.19.4.2 get_weight . . . . .	105
7.19.4.3 operator* . . . . .	105
7.19.4.4 operator++ . . . . .	105

7.19.4.5	<a href="#">operator++</a>	105
7.19.4.6	<a href="#">operator-</a>	105
7.19.4.7	<a href="#">operator-</a>	105
7.19.4.8	<a href="#">operator=</a>	106
7.19.5	<a href="#">Friends And Related Function Documentation</a>	106
7.19.5.1	<a href="#">operator!=</a>	106
7.19.5.2	<a href="#">operator==</a>	106
7.20	<a href="#">Xapian::ExpandDecider Class Reference</a>	107
7.20.1	<a href="#">Detailed Description</a>	107
7.20.2	<a href="#">Constructor &amp; Destructor Documentation</a>	107
7.20.2.1	<a href="#">~ExpandDecider</a>	107
7.20.3	<a href="#">Member Function Documentation</a>	107
7.20.3.1	<a href="#">operator()</a>	107
7.21	<a href="#">Xapian::ExpandDeciderAnd Class Reference</a>	108
7.21.1	<a href="#">Detailed Description</a>	108
7.21.2	<a href="#">Constructor &amp; Destructor Documentation</a>	108
7.21.2.1	<a href="#">ExpandDeciderAnd</a>	108
7.21.2.2	<a href="#">ExpandDeciderAnd</a>	109
7.21.3	<a href="#">Member Function Documentation</a>	109
7.21.3.1	<a href="#">operator()</a>	109
7.22	<a href="#">Xapian::ExpandDeciderFilterTerms Class Reference</a>	110
7.22.1	<a href="#">Detailed Description</a>	110
7.22.2	<a href="#">Constructor &amp; Destructor Documentation</a>	110
7.22.2.1	<a href="#">ExpandDeciderFilterTerms</a>	110
7.22.3	<a href="#">Member Function Documentation</a>	111
7.22.3.1	<a href="#">operator()</a>	111
7.23	<a href="#">Xapian::FeatureUnavailableError Class Reference</a>	112
7.23.1	<a href="#">Detailed Description</a>	112
7.23.2	<a href="#">Constructor &amp; Destructor Documentation</a>	112
7.23.2.1	<a href="#">FeatureUnavailableError</a>	112
7.23.2.2	<a href="#">FeatureUnavailableError</a>	113
7.24	<a href="#">Xapian::InternalError Class Reference</a>	114
7.24.1	<a href="#">Detailed Description</a>	114
7.24.2	<a href="#">Constructor &amp; Destructor Documentation</a>	114

7.24.2.1	InternalError	114
7.24.2.2	InternalError	115
7.25	Xapian::InvalidArgumentError Class Reference	116
7.25.1	Detailed Description	116
7.25.2	Constructor & Destructor Documentation	116
7.25.2.1	InvalidArgumentError	116
7.25.2.2	InvalidArgumentError	117
7.26	Xapian::InvalidOperationError Class Reference	118
7.26.1	Detailed Description	118
7.26.2	Constructor & Destructor Documentation	118
7.26.2.1	InvalidOperationError	118
7.26.2.2	InvalidOperationError	119
7.27	Xapian::LogicError Class Reference	120
7.27.1	Detailed Description	120
7.28	Xapian::MatchDecider Class Reference	121
7.28.1	Detailed Description	121
7.28.2	Constructor & Destructor Documentation	121
7.28.2.1	~MatchDecider	121
7.28.3	Member Function Documentation	121
7.28.3.1	operator()	121
7.29	Xapian::MSet Class Reference	122
7.29.1	Detailed Description	124
7.29.2	Member Typedef Documentation	124
7.29.2.1	const_iterator	124
7.29.2.2	const_reference	124
7.29.2.3	difference_type	124
7.29.2.4	iterator	125
7.29.2.5	pointer	125
7.29.2.6	reference	125
7.29.2.7	size_type	125
7.29.2.8	value_type	125
7.29.3	Constructor & Destructor Documentation	125
7.29.3.1	MSet	125
7.29.3.2	~MSet	125

7.29.3.3	MSet	125
7.29.4	Member Function Documentation	125
7.29.4.1	back	125
7.29.4.2	begin	125
7.29.4.3	convert_to_percent	126
7.29.4.4	convert_to_percent	126
7.29.4.5	empty	126
7.29.4.6	end	126
7.29.4.7	fetch	126
7.29.4.8	fetch	126
7.29.4.9	fetch	126
7.29.4.10	get_description	127
7.29.4.11	get_firstitem	127
7.29.4.12	get_matches_estimated	127
7.29.4.13	get_matches_lower_bound	127
7.29.4.14	get_matches_upper_bound	127
7.29.4.15	get_max_attained	127
7.29.4.16	get_max_possible	128
7.29.4.17	get_termfreq	128
7.29.4.18	get_termweight	128
7.29.4.19	max_size	128
7.29.4.20	operator=	128
7.29.4.21	operator[]	129
7.29.4.22	size	129
7.29.4.23	swap	129
7.30	Xapian::MSetIterator Class Reference	130
7.30.1	Detailed Description	131
7.30.2	Member Typedef Documentation	131
7.30.2.1	difference_type	131
7.30.2.2	iterator_category	132
7.30.2.3	pointer	132
7.30.2.4	reference	132
7.30.2.5	value_type	132
7.30.3	Constructor & Destructor Documentation	132

7.30.3.1	MSetIterator	132
7.30.3.2	MSetIterator	132
7.30.4	Member Function Documentation	132
7.30.4.1	get_collapse_count	132
7.30.4.2	get_collapse_key	133
7.30.4.3	get_description	133
7.30.4.4	get_document	133
7.30.4.5	get_percent	133
7.30.4.6	get_rank	133
7.30.4.7	get_weight	134
7.30.4.8	operator*	134
7.30.4.9	operator++	134
7.30.4.10	operator++	134
7.30.4.11	operator--	134
7.30.4.12	operator--	134
7.30.4.13	operator=	134
7.30.5	Friends And Related Function Documentation	134
7.30.5.1	operator!=	134
7.30.5.2	operator==	134
7.31	Xapian::MultiValueSorter Class Reference	136
7.31.1	Detailed Description	136
7.31.2	Member Function Documentation	136
7.31.2.1	operator()	136
7.32	Xapian::NetworkError Class Reference	137
7.32.1	Detailed Description	137
7.32.2	Constructor & Destructor Documentation	138
7.32.2.1	NetworkError	138
7.32.2.2	NetworkError	138
7.33	Xapian::NetworkTimeoutError Class Reference	139
7.33.1	Detailed Description	139
7.33.2	Constructor & Destructor Documentation	140
7.33.2.1	NetworkTimeoutError	140
7.33.2.2	NetworkTimeoutError	140
7.34	Xapian::NumberValueRangeProcessor Class Reference	141



---

7.34.1	Detailed Description	141
7.34.2	Constructor & Destructor Documentation	141
7.34.2.1	NumberValueRangeProcessor	141
7.34.2.2	NumberValueRangeProcessor	142
7.34.3	Member Function Documentation	142
7.34.3.1	operator()	142
7.35	Xapian::PositionIterator Class Reference	143
7.35.1	Detailed Description	144
7.35.2	Constructor & Destructor Documentation	144
7.35.2.1	PositionIterator	144
7.35.2.2	~PositionIterator	144
7.35.2.3	PositionIterator	144
7.35.3	Member Function Documentation	144
7.35.3.1	get_description	144
7.35.3.2	operator=	144
7.35.3.3	skip_to	144
7.35.4	Friends And Related Function Documentation	144
7.35.4.1	operator==	144
7.36	Xapian::PostingIterator Class Reference	146
7.36.1	Detailed Description	147
7.36.2	Member Typedef Documentation	147
7.36.2.1	difference_type	147
7.36.2.2	iterator_category	147
7.36.2.3	pointer	147
7.36.2.4	reference	147
7.36.2.5	value_type	148
7.36.3	Constructor & Destructor Documentation	148
7.36.3.1	PostingIterator	148
7.36.3.2	~PostingIterator	148
7.36.3.3	PostingIterator	148
7.36.4	Member Function Documentation	148
7.36.4.1	get_description	148
7.36.4.2	get_doclength	148
7.36.4.3	get_wdf	148

7.36.4.4	<a href="#">operator*</a>	148
7.36.4.5	<a href="#">operator=</a>	149
7.36.4.6	<a href="#">positionlist_begin</a>	149
7.36.4.7	<a href="#">positionlist_end</a>	149
7.36.4.8	<a href="#">skip_to</a>	149
7.36.5	<a href="#">Friends And Related Function Documentation</a>	149
7.36.5.1	<a href="#">operator==</a>	149
7.37	<a href="#">Xapian::Query Class Reference</a>	150
7.37.1	<a href="#">Detailed Description</a>	151
7.37.2	<a href="#">Member Enumeration Documentation</a>	152
7.37.2.1	<a href="#">op</a>	152
7.37.3	<a href="#">Constructor &amp; Destructor Documentation</a>	152
7.37.3.1	<a href="#">Query</a>	152
7.37.3.2	<a href="#">Query</a>	153
7.37.3.3	<a href="#">~Query</a>	153
7.37.3.4	<a href="#">Query</a>	153
7.37.3.5	<a href="#">Query</a>	153
7.37.3.6	<a href="#">Query</a>	153
7.37.3.7	<a href="#">Query</a>	153
7.37.3.8	<a href="#">Query</a>	153
7.37.3.9	<a href="#">Query</a>	154
7.37.3.10	<a href="#">Query</a>	154
7.37.3.11	<a href="#">Query</a>	154
7.37.4	<a href="#">Member Function Documentation</a>	154
7.37.4.1	<a href="#">empty</a>	154
7.37.4.2	<a href="#">get_description</a>	155
7.37.4.3	<a href="#">get_length</a>	155
7.37.4.4	<a href="#">get_terms_begin</a>	155
7.37.4.5	<a href="#">get_terms_end</a>	155
7.37.4.6	<a href="#">operator=</a>	155
7.37.5	<a href="#">Member Data Documentation</a>	155
7.37.5.1	<a href="#">MatchAll</a>	155
7.37.5.2	<a href="#">MatchNothing</a>	155
7.38	<a href="#">Xapian::QueryParser Class Reference</a>	156

7.38.1 Detailed Description . . . . .	157
7.38.2 Member Enumeration Documentation . . . . .	157
7.38.2.1 feature_flag . . . . .	157
7.38.3 Constructor & Destructor Documentation . . . . .	159
7.38.3.1 QueryParser . . . . .	159
7.38.3.2 QueryParser . . . . .	159
7.38.3.3 ~QueryParser . . . . .	159
7.38.4 Member Function Documentation . . . . .	159
7.38.4.1 add_boolean_prefix . . . . .	159
7.38.4.2 add_prefix . . . . .	160
7.38.4.3 add_valuerangeprocessor . . . . .	161
7.38.4.4 get_corrected_query_string . . . . .	161
7.38.4.5 get_default_op . . . . .	161
7.38.4.6 get_description . . . . .	161
7.38.4.7 operator= . . . . .	161
7.38.4.8 parse_query . . . . .	161
7.38.4.9 set_database . . . . .	162
7.38.4.10 set_default_op . . . . .	162
7.38.4.11 set_stemmer . . . . .	162
7.38.4.12 set_stemming_strategy . . . . .	163
7.38.4.13 set_stopper . . . . .	163
7.38.4.14 stoplist_begin . . . . .	163
7.38.4.15 unstem_begin . . . . .	163
7.39 Xapian::QueryParserError Class Reference . . . . .	164
7.39.1 Detailed Description . . . . .	164
7.39.2 Constructor & Destructor Documentation . . . . .	164
7.39.2.1 QueryParserError . . . . .	164
7.39.2.2 QueryParserError . . . . .	165
7.40 Xapian::RangeError Class Reference . . . . .	166
7.40.1 Detailed Description . . . . .	166
7.40.2 Constructor & Destructor Documentation . . . . .	166
7.40.2.1 RangeError . . . . .	166
7.40.2.2 RangeError . . . . .	167
7.41 Xapian::RSet Class Reference . . . . .	168

7.41.1	Detailed Description	169
7.41.2	Constructor & Destructor Documentation	169
7.41.2.1	RSet	169
7.41.2.2	RSet	169
7.41.2.3	~RSet	169
7.41.3	Member Function Documentation	169
7.41.3.1	add_document	169
7.41.3.2	add_document	169
7.41.3.3	contains	169
7.41.3.4	contains	170
7.41.3.5	empty	170
7.41.3.6	get_description	170
7.41.3.7	operator=	170
7.41.3.8	remove_document	170
7.41.3.9	remove_document	170
7.41.3.10	size	170
7.42	Xapian::RuntimeError Class Reference	171
7.42.1	Detailed Description	171
7.43	Xapian::SimpleStopper Class Reference	172
7.43.1	Detailed Description	172
7.43.2	Constructor & Destructor Documentation	173
7.43.2.1	SimpleStopper	173
7.43.2.2	SimpleStopper	173
7.43.2.3	~SimpleStopper	173
7.43.3	Member Function Documentation	173
7.43.3.1	add	173
7.43.3.2	get_description	173
7.43.3.3	operator()	173
7.44	Xapian::Sorter Class Reference	174
7.44.1	Detailed Description	174
7.44.2	Constructor & Destructor Documentation	174
7.44.2.1	~Sorter	174
7.44.3	Member Function Documentation	174
7.44.3.1	operator()	174

7.45 Xapian::Stem Class Reference . . . . .	176
7.45.1 Detailed Description . . . . .	176
7.45.2 Constructor & Destructor Documentation . . . . .	176
7.45.2.1 Stem . . . . .	176
7.45.2.2 Stem . . . . .	177
7.45.2.3 Stem . . . . .	177
7.45.2.4 ~Stem . . . . .	178
7.45.3 Member Function Documentation . . . . .	178
7.45.3.1 get_available_languages . . . . .	178
7.45.3.2 get_description . . . . .	178
7.45.3.3 operator() . . . . .	178
7.45.3.4 operator= . . . . .	178
7.46 Xapian::Stopper Class Reference . . . . .	179
7.46.1 Detailed Description . . . . .	179
7.46.2 Constructor & Destructor Documentation . . . . .	179
7.46.2.1 ~Stopper . . . . .	179
7.46.3 Member Function Documentation . . . . .	179
7.46.3.1 get_description . . . . .	179
7.46.3.2 operator() . . . . .	180
7.47 Xapian::StringValueRangeProcessor Class Reference . . . . .	181
7.47.1 Detailed Description . . . . .	181
7.47.2 Constructor & Destructor Documentation . . . . .	181
7.47.2.1 StringValueRangeProcessor . . . . .	181
7.47.3 Member Function Documentation . . . . .	182
7.47.3.1 operator() . . . . .	182
7.48 Xapian::TermGenerator Class Reference . . . . .	183
7.48.1 Detailed Description . . . . .	184
7.48.2 Member Enumeration Documentation . . . . .	184
7.48.2.1 flags . . . . .	184
7.48.3 Constructor & Destructor Documentation . . . . .	185
7.48.3.1 TermGenerator . . . . .	185
7.48.3.2 TermGenerator . . . . .	185
7.48.3.3 ~TermGenerator . . . . .	185
7.48.4 Member Function Documentation . . . . .	185

7.48.4.1	<a href="#">get_description</a>	185
7.48.4.2	<a href="#">get_document</a>	185
7.48.4.3	<a href="#">get_termpos</a>	185
7.48.4.4	<a href="#">increase_termpos</a>	185
7.48.4.5	<a href="#">index_text</a>	185
7.48.4.6	<a href="#">index_text</a>	186
7.48.4.7	<a href="#">index_text_without_positions</a>	186
7.48.4.8	<a href="#">index_text_without_positions</a>	186
7.48.4.9	<a href="#">operator=</a>	186
7.48.4.10	<a href="#">set_database</a>	186
7.48.4.11	<a href="#">set_document</a>	186
7.48.4.12	<a href="#">set_flags</a>	187
7.48.4.13	<a href="#">set_stemmer</a>	187
7.48.4.14	<a href="#">set_stopper</a>	187
7.48.4.15	<a href="#">set_termpos</a>	187
7.49	<a href="#">Xapian::TermIterator Class Reference</a>	188
7.49.1	<a href="#">Detailed Description</a>	189
7.49.2	<a href="#">Member Typedef Documentation</a>	189
7.49.2.1	<a href="#">difference_type</a>	189
7.49.2.2	<a href="#">iterator_category</a>	189
7.49.2.3	<a href="#">pointer</a>	189
7.49.2.4	<a href="#">reference</a>	189
7.49.2.5	<a href="#">value_type</a>	190
7.49.3	<a href="#">Constructor &amp; Destructor Documentation</a>	190
7.49.3.1	<a href="#">TermIterator</a>	190
7.49.3.2	<a href="#">~TermIterator</a>	190
7.49.3.3	<a href="#">TermIterator</a>	190
7.49.4	<a href="#">Member Function Documentation</a>	190
7.49.4.1	<a href="#">get_description</a>	190
7.49.4.2	<a href="#">get_termfreq</a>	190
7.49.4.3	<a href="#">get_wdf</a>	190
7.49.4.4	<a href="#">operator*</a>	190
7.49.4.5	<a href="#">operator=</a>	191
7.49.4.6	<a href="#">positionlist_begin</a>	191

7.49.4.7	<a href="#">positionlist_count</a>	191
7.49.4.8	<a href="#">positionlist_end</a>	191
7.49.4.9	<a href="#">skip_to</a>	191
7.50	<a href="#">Xapian::TradWeight Class Reference</a>	192
7.50.1	<a href="#">Detailed Description</a>	193
7.50.2	<a href="#">Constructor &amp; Destructor Documentation</a>	193
7.50.2.1	<a href="#">TradWeight</a>	193
7.50.3	<a href="#">Member Function Documentation</a>	193
7.50.3.1	<a href="#">clone</a>	193
7.50.3.2	<a href="#">get_maxextra</a>	194
7.50.3.3	<a href="#">get_maxpart</a>	194
7.50.3.4	<a href="#">get_sumextra</a>	194
7.50.3.5	<a href="#">get_sumpart</a>	194
7.50.3.6	<a href="#">get_sumpart_needs_doclength</a>	195
7.50.3.7	<a href="#">name</a>	195
7.50.3.8	<a href="#">serialise</a>	195
7.50.3.9	<a href="#">unserialise</a>	195
7.51	<a href="#">Xapian::UnimplementedError Class Reference</a>	196
7.51.1	<a href="#">Detailed Description</a>	196
7.51.2	<a href="#">Constructor &amp; Destructor Documentation</a>	196
7.51.2.1	<a href="#">UnimplementedError</a>	196
7.51.2.2	<a href="#">UnimplementedError</a>	197
7.52	<a href="#">Xapian::Utf8Iterator Class Reference</a>	198
7.52.1	<a href="#">Detailed Description</a>	199
7.52.2	<a href="#">Member Typedef Documentation</a>	199
7.52.2.1	<a href="#">difference_type</a>	199
7.52.2.2	<a href="#">iterator_category</a>	199
7.52.2.3	<a href="#">pointer</a>	199
7.52.2.4	<a href="#">reference</a>	199
7.52.2.5	<a href="#">value_type</a>	199
7.52.3	<a href="#">Constructor &amp; Destructor Documentation</a>	200
7.52.3.1	<a href="#">Utf8Iterator</a>	200
7.52.3.2	<a href="#">Utf8Iterator</a>	200
7.52.3.3	<a href="#">Utf8Iterator</a>	200

7.52.3.4	Utf8Iterator	200
7.52.4	Member Function Documentation	201
7.52.4.1	assign	201
7.52.4.2	assign	201
7.52.4.3	left	201
7.52.4.4	operator!=	201
7.52.4.5	operator*	201
7.52.4.6	operator++	202
7.52.4.7	operator++	202
7.52.4.8	operator==	202
7.52.4.9	raw	202
7.53	Xapian::ValueIterator Class Reference	203
7.53.1	Detailed Description	204
7.53.2	Member Typedef Documentation	204
7.53.2.1	difference_type	204
7.53.2.2	iterator_category	204
7.53.2.3	pointer	204
7.53.2.4	reference	204
7.53.2.5	value_type	204
7.53.3	Constructor & Destructor Documentation	205
7.53.3.1	ValueIterator	205
7.53.3.2	ValueIterator	205
7.53.4	Member Function Documentation	205
7.53.4.1	get_description	205
7.53.4.2	get_valueno	205
7.53.4.3	operator*	205
7.53.4.4	operator++	205
7.53.4.5	operator++	205
7.53.4.6	operator->	205
7.53.4.7	operator=	205
7.53.5	Friends And Related Function Documentation	206
7.53.5.1	operator!=	206
7.53.5.2	operator==	206
7.54	Xapian::ValueRangeProcessor Struct Reference	207



7.54.1 Detailed Description . . . . .	207
7.54.2 Constructor & Destructor Documentation . . . . .	207
7.54.2.1 ~ValueRangeProcessor . . . . .	207
7.54.3 Member Function Documentation . . . . .	207
7.54.3.1 operator() . . . . .	207
7.55 Xapian::Weight Class Reference . . . . .	209
7.55.1 Detailed Description . . . . .	210
7.55.2 Member Function Documentation . . . . .	210
7.55.2.1 create . . . . .	210
7.55.2.2 get_maxextra . . . . .	210
7.55.2.3 get_maxpart . . . . .	211
7.55.2.4 get_sumextra . . . . .	211
7.55.2.5 get_sumpart . . . . .	211
7.55.2.6 get_sumpart_needs_doclength . . . . .	211
7.55.2.7 name . . . . .	212
7.55.2.8 serialise . . . . .	212
7.55.2.9 unserialise . . . . .	212
7.56 Xapian::WritableDatabase Class Reference . . . . .	213
7.56.1 Detailed Description . . . . .	214
7.56.2 Constructor & Destructor Documentation . . . . .	215
7.56.2.1 ~WritableDatabase . . . . .	215
7.56.2.2 WritableDatabase . . . . .	215
7.56.2.3 WritableDatabase . . . . .	215
7.56.2.4 WritableDatabase . . . . .	215
7.56.3 Member Function Documentation . . . . .	216
7.56.3.1 add_document . . . . .	216
7.56.3.2 add_spelling . . . . .	216
7.56.3.3 add_synonym . . . . .	216
7.56.3.4 begin_transaction . . . . .	217
7.56.3.5 cancel_transaction . . . . .	217
7.56.3.6 clear_synonyms . . . . .	218
7.56.3.7 commit_transaction . . . . .	218
7.56.3.8 delete_document . . . . .	218
7.56.3.9 delete_document . . . . .	219

7.56.3.10 flush . . . . .	219
7.56.3.11 get_description . . . . .	220
7.56.3.12 operator= . . . . .	220
7.56.3.13 remove_spelling . . . . .	220
7.56.3.14 remove_synonym . . . . .	220
7.56.3.15 replace_document . . . . .	221
7.56.3.16 replace_document . . . . .	221
7.56.3.17 set_metadata . . . . .	222
<b>8 File Documentation</b>	<b>225</b>
8.1 xapian/error.h File Reference . . . . .	225
8.1.1 Detailed Description . . . . .	227
8.2 xapian/version.h File Reference . . . . .	228
8.2.1 Detailed Description . . . . .	228
8.2.2 Define Documentation . . . . .	228
8.2.2.1 XAPIAN_HAS_FLINT_BACKEND . . . . .	228
8.2.2.2 XAPIAN_HAS_INMEMORY_BACKEND . . . . .	229
8.2.2.3 XAPIAN_HAS_QUARTZ_BACKEND . . . . .	229
8.2.2.4 XAPIAN_HAS_REMOTE_BACKEND . . . . .	229
8.2.2.5 XAPIAN_MAJOR_VERSION . . . . .	229
8.2.2.6 XAPIAN_MINOR_VERSION . . . . .	229
8.2.2.7 XAPIAN_REVISION . . . . .	229
8.2.2.8 XAPIAN_VERSION . . . . .	229
8.3 xapian.h File Reference . . . . .	230
8.3.1 Detailed Description . . . . .	230
8.4 xapian/database.h File Reference . . . . .	231
8.4.1 Detailed Description . . . . .	231
8.5 xapian/dbfactory.h File Reference . . . . .	232
8.5.1 Detailed Description . . . . .	233
8.6 xapian/document.h File Reference . . . . .	234
8.6.1 Detailed Description . . . . .	234
8.7 xapian/enquire.h File Reference . . . . .	235
8.7.1 Detailed Description . . . . .	236
8.8 xapian/errorhandler.h File Reference . . . . .	237

8.8.1 Detailed Description . . . . .	237
8.9 xapian/expanddecider.h File Reference . . . . .	238
8.9.1 Detailed Description . . . . .	238
8.10 xapian/positioniterator.h File Reference . . . . .	239
8.10.1 Detailed Description . . . . .	239
8.11 xapian/postingiterator.h File Reference . . . . .	240
8.11.1 Detailed Description . . . . .	240
8.12 xapian/query.h File Reference . . . . .	241
8.12.1 Detailed Description . . . . .	241
8.13 xapian/queryparser.h File Reference . . . . .	242
8.13.1 Detailed Description . . . . .	243
8.14 xapian/sorter.h File Reference . . . . .	244
8.14.1 Detailed Description . . . . .	244
8.15 xapian/stem.h File Reference . . . . .	245
8.15.1 Detailed Description . . . . .	245
8.16 xapian/termgenerator.h File Reference . . . . .	246
8.16.1 Detailed Description . . . . .	246
8.17 xapian/termiterator.h File Reference . . . . .	247
8.17.1 Detailed Description . . . . .	247
8.18 xapian/types.h File Reference . . . . .	248
8.18.1 Detailed Description . . . . .	249
8.19 xapian/unicode.h File Reference . . . . .	250
8.19.1 Detailed Description . . . . .	251
8.20 xapian/valueiterator.h File Reference . . . . .	252
8.20.1 Detailed Description . . . . .	252



# Chapter 1

## Deprecated List

**Member [Xapian::Enquire::register\\_match\\_decider](#)**(const std::string &name, const MatchDecider \*mdecider=NULL)

This method is deprecated. It was added long ago with the intention that it would allow the remote backend to support use of MatchDecider objects, but there's a better approach.

**Member [Xapian::Error::get\\_errno](#)**() const This method is deprecated, because errno values aren't portable between platforms, so we can't serialise them when passing exceptions from a remote server to a client. Use the [get\\_error\\_string](#)() method instead.

**Member [Xapian::Query::Query](#)**(Query::op op\_, [Xapian::Query](#) q) This method is deprecated because it isn't useful, since none of the current query operators can be usefully applied to a single subquery without a parameter value.

**Member [Xapian::xapian\\_major\\_version](#)**() This function is now deprecated, use [Xapian::major\\_version](#)() instead.

**Member [Xapian::xapian\\_minor\\_version](#)**() This function is now deprecated, use [Xapian::minor\\_version](#)() instead.

**Member [Xapian::xapian\\_revision](#)**() This function is now deprecated, use [Xapian::revision](#)() instead.

**Member [Xapian::xapian\\_version\\_string](#)**() This function is now deprecated, use [Xapian::version\\_string](#)() instead.



## Chapter 2

# Namespace Index

### 2.1 Namespace List

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

<a href="#">Xapian</a> (The <a href="#">Xapian</a> namespace contains public interfaces for the <a href="#">Xapian</a> library ) . . . . .	13
<a href="#">Xapian::Auto</a> ( <a href="#">Database</a> factory functions which determine the database type automatically ) . . . . .	26
<a href="#">Xapian::Flint</a> ( <a href="#">Database</a> factory functions for the flint backend ) . . . . .	27
<a href="#">Xapian::InMemory</a> ( <a href="#">Database</a> factory functions for the inmemory backend ) .	29
<a href="#">Xapian::Quartz</a> ( <a href="#">Database</a> factory functions for the quartz backend ) . . . . .	30
<a href="#">Xapian::Remote</a> ( <a href="#">Database</a> factory functions for the remote backend ) . . . .	32
<a href="#">Xapian::Unicode</a> (Functions associated with handling <a href="#">Unicode</a> characters ) .	35





## Chapter 3

# Class Index

### 3.1 Class Hierarchy

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

Xapian::Database . . . . .	48
Xapian::WritableDatabase . . . . .	213
Xapian::Document . . . . .	75
Xapian::Enquire . . . . .	82
Xapian::Error . . . . .	95
Xapian::LogicError . . . . .	120
Xapian::AssertionError . . . . .	39
Xapian::InvalidArgumentError . . . . .	116
Xapian::InvalidOperationError . . . . .	118
Xapian::UnimplementedError . . . . .	196
Xapian::RuntimeError . . . . .	171
Xapian::DatabaseError . . . . .	62
Xapian::DatabaseCorruptError . . . . .	58
Xapian::DatabaseCreateError . . . . .	60
Xapian::DatabaseLockError . . . . .	63
Xapian::DatabaseModifiedError . . . . .	65
Xapian::DatabaseOpeningError . . . . .	67
Xapian::DatabaseVersionError . . . . .	69
Xapian::DocNotFoundError . . . . .	73
Xapian::FeatureUnavailableError . . . . .	112
Xapian::InternalError . . . . .	114
Xapian::NetworkError . . . . .	137
Xapian::NetworkTimeoutError . . . . .	139
Xapian::QueryParserError . . . . .	164
Xapian::RangeError . . . . .	166
Xapian::ErrorHandler . . . . .	98
Xapian::ESet . . . . .	100
Xapian::ESetIterator . . . . .	103

Xapian::ExpandDecider . . . . .	107
Xapian::ExpandDeciderAnd . . . . .	108
Xapian::ExpandDeciderFilterTerms . . . . .	110
Xapian::MatchDecider . . . . .	121
Xapian::MSet . . . . .	122
Xapian::MSetIterator . . . . .	130
Xapian::PositionIterator . . . . .	143
Xapian::PostingIterator . . . . .	146
Xapian::Query . . . . .	150
Xapian::QueryParser . . . . .	156
Xapian::RSet . . . . .	168
Xapian::Sorter . . . . .	174
Xapian::MultiValueSorter . . . . .	136
Xapian::Stem . . . . .	176
Xapian::Stopper . . . . .	179
Xapian::SimpleStopper . . . . .	172
Xapian::TermGenerator . . . . .	183
Xapian::TermIterator . . . . .	188
Xapian::Utf8Iterator . . . . .	198
Xapian::ValueIterator . . . . .	203
Xapian::ValueRangeProcessor . . . . .	207
Xapian::DateValueRangeProcessor . . . . .	71
Xapian::NumberValueRangeProcessor . . . . .	141
Xapian::StringValueRangeProcessor . . . . .	181
Xapian::Weight . . . . .	209
Xapian::BM25Weight . . . . .	41
Xapian::BoolWeight . . . . .	45
Xapian::TradWeight . . . . .	192

## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">Xapian::AssertionError</a> ( <a href="#">AssertionError</a> is thrown if a logical assertion inside <a href="#">Xapian</a> fails ) . . . . .	39
<a href="#">Xapian::BM25Weight</a> (BM25 weighting scheme ) . . . . .	41
<a href="#">Xapian::BoolWeight</a> (Boolean weighting scheme (everything gets 0) ) . . . .	45
<a href="#">Xapian::Database</a> (This class is used to access a database, or a group of databases ) . . . . .	48
<a href="#">Xapian::DatabaseCorruptError</a> ( <a href="#">DatabaseCorruptError</a> indicates database corruption was detected ) . . . . .	58
<a href="#">Xapian::DatabaseCreateError</a> ( <a href="#">DatabaseCreateError</a> indicates a failure to create a database ) . . . . .	60
<a href="#">Xapian::DatabaseError</a> ( <a href="#">DatabaseError</a> indicates some sort of database related error ) . . . . .	62
<a href="#">Xapian::DatabaseLockError</a> ( <a href="#">DatabaseLockError</a> indicates failure to lock a database ) . . . . .	63
<a href="#">Xapian::DatabaseModifiedError</a> ( <a href="#">DatabaseModifiedError</a> indicates a database was modified ) . . . . .	65
<a href="#">Xapian::DatabaseOpeningError</a> ( <a href="#">DatabaseOpeningError</a> indicates failure to open a database ) . . . . .	67
<a href="#">Xapian::DatabaseVersionError</a> ( <a href="#">DatabaseVersionError</a> indicates that a database is in an unsupported format ) . . . . .	69
<a href="#">Xapian::DateValueRangeProcessor</a> (Handle a date range ) . . . . .	71
<a href="#">Xapian::DocNotFoundError</a> (Indicates an attempt to access a document not present in the database ) . . . . .	73
<a href="#">Xapian::Document</a> (A handle representing a document in a <a href="#">Xapian</a> database )	75
<a href="#">Xapian::Enquire</a> (This class provides an interface to the information retrieval system for the purpose of searching ) . . . . .	82
<a href="#">Xapian::Error</a> (All exceptions thrown by <a href="#">Xapian</a> are subclasses of <a href="#">Xapian::Error</a> ) . . . . .	95

<a href="#">Xapian::ErrorHandler</a> (Decide if a <a href="#">Xapian::Error</a> exception should be ignored) . . . . .	98
<a href="#">Xapian::ESet</a> (Class representing an ordered set of expand terms (an <a href="#">ESet</a> ) ) . . . . .	100
<a href="#">Xapian::ESetIterator</a> (Iterate through terms in the <a href="#">ESet</a> ) . . . . .	103
<a href="#">Xapian::ExpandDecider</a> (Virtual base class for expand decider functor) . . . . .	107
<a href="#">Xapian::ExpandDeciderAnd</a> ( <a href="#">ExpandDecider</a> subclass which rejects terms using two <a href="#">ExpandDeciders</a> ) . . . . .	108
<a href="#">Xapian::ExpandDeciderFilterTerms</a> ( <a href="#">ExpandDecider</a> subclass which rejects terms in a specified list) . . . . .	110
<a href="#">Xapian::FeatureUnavailableError</a> (Indicates an attempt to use a feature which is unavailable) . . . . .	112
<a href="#">Xapian::InternalError</a> ( <a href="#">InternalError</a> indicates a runtime problem of some sort) . . . . .	114
<a href="#">Xapian::InvalidArgumentError</a> ( <a href="#">InvalidArgumentError</a> indicates an invalid parameter value was passed to the API) . . . . .	116
<a href="#">Xapian::InvalidOperationError</a> ( <a href="#">InvalidOperationError</a> indicates the API was used in an invalid way) . . . . .	118
<a href="#">Xapian::LogicError</a> (The base class for exceptions indicating errors in the program logic) . . . . .	120
<a href="#">Xapian::MatchDecider</a> (Base class for matcher decision functor) . . . . .	121
<a href="#">Xapian::MSet</a> (A match set ( <a href="#">MSet</a> )) . . . . .	122
<a href="#">Xapian::MSetIterator</a> (An iterator pointing to items in an <a href="#">MSet</a> ) . . . . .	130
<a href="#">Xapian::MultiValueSorter</a> ( <a href="#">Sorter</a> subclass which sorts by a several values) . . . . .	136
<a href="#">Xapian::NetworkError</a> (Indicates a problem communicating with a remote database) . . . . .	137
<a href="#">Xapian::NetworkTimeoutError</a> (Indicates a timeout expired while communicating with a remote database) . . . . .	139
<a href="#">Xapian::NumberValueRangeProcessor</a> (Handle a number range) . . . . .	141
<a href="#">Xapian::PositionIterator</a> (An iterator pointing to items in a list of positions) . . . . .	143
<a href="#">Xapian::PostingIterator</a> (An iterator pointing to items in a list of postings) . . . . .	146
<a href="#">Xapian::Query</a> (Class representing a query) . . . . .	150
<a href="#">Xapian::QueryParser</a> (Build a <a href="#">Xapian::Query</a> object from a user query string) . . . . .	156
<a href="#">Xapian::QueryParserError</a> (Indicates a query string can't be parsed) . . . . .	164
<a href="#">Xapian::RangeError</a> ( <a href="#">RangeError</a> indicates an attempt to access outside the bounds of a container) . . . . .	166
<a href="#">Xapian::RSet</a> (A relevance set (R-Set)) . . . . .	168
<a href="#">Xapian::RuntimeError</a> (The base class for exceptions indicating errors only detectable at runtime) . . . . .	171
<a href="#">Xapian::SimpleStopper</a> (Simple implementation of <a href="#">Stopper</a> class - this will suit most users) . . . . .	172
<a href="#">Xapian::Sorter</a> (Virtual base class for sorter functor) . . . . .	174
<a href="#">Xapian::Stem</a> (Class representing a stemming algorithm) . . . . .	176
<a href="#">Xapian::Stopper</a> (Base class for stop-word decision functor) . . . . .	179
<a href="#">Xapian::StringValueRangeProcessor</a> (Handle a string range) . . . . .	181
<a href="#">Xapian::TermGenerator</a> (Parses a piece of text and generate terms) . . . . .	183
<a href="#">Xapian::TermIterator</a> (An iterator pointing to items in a list of terms) . . . . .	188
<a href="#">Xapian::TradWeight</a> (Traditional probabilistic weighting scheme) . . . . .	192
<a href="#">Xapian::UnimplementedError</a> ( <a href="#">UnimplementedError</a> indicates an attempt to use an unimplemented feature) . . . . .	196

---

<a href="#">Xapian::Utf8Iterator</a> (An iterator which returns <a href="#">Unicode</a> character values from a UTF-8 encoded string ) . . . . .	198
<a href="#">Xapian::ValueIterator</a> (An iterator pointing to values associated with a document ) . . . . .	203
<a href="#">Xapian::ValueRangeProcessor</a> (Base class for value range processors ) . . . .	207
<a href="#">Xapian::Weight</a> (Abstract base class for weighting schemes ) . . . . .	209
<a href="#">Xapian::WritableDatabase</a> (This class provides read/write access to a database ) . . . . .	213



## Chapter 5

# File Index

### 5.1 File List

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

xapian/error.h (Hierarchy of classes which <a href="#">Xapian</a> can throw as exceptions )	225
xapian/version.h (Define preprocessor symbols for the library version )	228
<a href="#">xapian.h</a> (Public interfaces for the <a href="#">Xapian</a> library )	230
xapian/database.h (API for working with <a href="#">Xapian</a> databases )	231
xapian/dbfactory.h (Factory functions for constructing Database and WritableDatabase objects )	232
xapian/document.h (API for working with documents )	234
xapian/enquire.h (API for running queries )	235
xapian/errorhandler.h (Decide if a <a href="#">Xapian::Error</a> exception should be ignored )	237
xapian/expanddecider.h (Allow rejection of terms during ESet generation )	238
xapian/positioniterator.h (Classes for iterating through position lists )	239
xapian/postingiterator.h (Classes for iterating through posting lists )	240
xapian/query.h (Classes for representing a query )	241
xapian/queryparser.h (Parsing a user query string to build a <a href="#">Xapian::Query</a> object )	242
xapian/sorter.h (Build sort keys for MSet ordering )	244
xapian/stem.h (Stemming algorithms )	245
xapian/termgenerator.h (Parse free text and generate terms )	246
xapian/termiterator.h (Classes for iterating through term lists )	247
xapian/types.h (Typedefs for <a href="#">Xapian</a> )	248
xapian/unicode.h (Unicode and UTF-8 related classes and functions )	250
xapian/valueiterator.h (Classes for iterating through values )	252





## Chapter 6

# Namespace Documentation

### 6.1 Xapian Namespace Reference

The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.

#### Namespaces

- namespace [Auto](#)  
*[Database](#) factory functions which determine the database type automatically.*
- namespace [Flint](#)  
*[Database](#) factory functions for the flint backend.*
- namespace [InMemory](#)  
*[Database](#) factory functions for the inmemory backend.*
- namespace [Quartz](#)  
*[Database](#) factory functions for the quartz backend.*
- namespace [Remote](#)  
*[Database](#) factory functions for the remote backend.*
- namespace [Unicode](#)  
*Functions associated with handling [Unicode](#) characters.*

#### Classes

- class [Error](#)  
*All exceptions thrown by [Xapian](#) are subclasses of [Xapian::Error](#).*

- class [LogicError](#)  
*The base class for exceptions indicating errors in the program logic.*
- class [RuntimeError](#)  
*The base class for exceptions indicating errors only detectable at runtime.*
- class [AssertionError](#)  
*[AssertionError](#) is thrown if a logical assertion inside [Xapian](#) fails.*
- class [InvalidArgumentError](#)  
*[InvalidArgumentError](#) indicates an invalid parameter value was passed to the API.*
- class [InvalidOperationError](#)  
*[InvalidOperationError](#) indicates the API was used in an invalid way.*
- class [UnimplementedError](#)  
*[UnimplementedError](#) indicates an attempt to use an unimplemented feature.*
- class [DatabaseError](#)  
*[DatabaseError](#) indicates some sort of database related error.*
- class [DatabaseCorruptError](#)  
*[DatabaseCorruptError](#) indicates database corruption was detected.*
- class [DatabaseCreateError](#)  
*[DatabaseCreateError](#) indicates a failure to create a database.*
- class [DatabaseLockError](#)  
*[DatabaseLockError](#) indicates failure to lock a database.*
- class [DatabaseModifiedError](#)  
*[DatabaseModifiedError](#) indicates a database was modified.*
- class [DatabaseOpeningError](#)  
*[DatabaseOpeningError](#) indicates failure to open a database.*
- class [DatabaseVersionError](#)  
*[DatabaseVersionError](#) indicates that a database is in an unsupported format.*
- class [DocNotFoundError](#)  
*Indicates an attempt to access a document not present in the database.*
- class [FeatureUnavailableError](#)  
*Indicates an attempt to use a feature which is unavailable.*
- class [InternalError](#)

*[InternalError](#) indicates a runtime problem of some sort.*

- class [NetworkError](#)

*Indicates a problem communicating with a remote database.*

- class [NetworkTimeoutError](#)

*Indicates a timeout expired while communicating with a remote database.*

- class [QueryParserError](#)

*Indicates a query string can't be parsed.*

- class [RangeError](#)

*[RangeError](#) indicates an attempt to access outside the bounds of a container.*

- class [Database](#)

*This class is used to access a database, or a group of databases.*

- class [WritableDatabase](#)

*This class provides read/write access to a database.*

- class [Document](#)

*A handle representing a document in a [Xapian](#) database.*

- class [MSet](#)

*A match set ([MSet](#)).*

- class [MSetIterator](#)

*An iterator pointing to items in an [MSet](#).*

- class [ESet](#)

*Class representing an ordered set of expand terms (an [ESet](#)).*

- class [ESetIterator](#)

*Iterate through terms in the [ESet](#).*

- class [RSet](#)

*A relevance set (R-Set).*

- class [MatchDecider](#)

*Base class for matcher decision functor.*

- class [Enquire](#)

*This class provides an interface to the information retrieval system for the purpose of searching.*

- class [Weight](#)

*Abstract base class for weighting schemes.*

- class [BoolWeight](#)  
*Boolean weighting scheme (everything gets 0).*
- class [BM25Weight](#)  
*BM25 weighting scheme.*
- class [TradWeight](#)  
*Traditional probabilistic weighting scheme.*
- class [ErrorHandler](#)  
*Decide if a [Xapian::Error](#) exception should be ignored.*
- class [ExpandDecider](#)  
*Virtual base class for expand decider functor.*
- class [ExpandDeciderAnd](#)  
*[ExpandDecider](#) subclass which rejects terms using two [ExpandDeciders](#).*
- class [ExpandDeciderFilterTerms](#)  
*[ExpandDecider](#) subclass which rejects terms in a specified list.*
- class [PositionIterator](#)  
*An iterator pointing to items in a list of positions.*
- class [PostingIterator](#)  
*An iterator pointing to items in a list of postings.*
- class [Query](#)  
*Class representing a query.*
- class [Stopper](#)  
*Base class for stop-word decision functor.*
- class [SimpleStopper](#)  
*Simple implementation of [Stopper](#) class - this will suit most users.*
- struct [ValueRangeProcessor](#)  
*Base class for value range processors.*
- class [StringValueRangeProcessor](#)  
*Handle a string range.*
- class [DateValueRangeProcessor](#)  
*Handle a date range.*

- class [NumberValueRangeProcessor](#)  
*Handle a number range.*
- class [QueryParser](#)  
*Build a [Xapian::Query](#) object from a user query string.*
- class [Sorter](#)  
*Virtual base class for sorter functor.*
- class [MultiValueSorter](#)  
*[Sorter](#) subclass which sorts by a several values.*
- class [Stem](#)  
*Class representing a stemming algorithm.*
- class [TermGenerator](#)  
*Parses a piece of text and generate terms.*
- class [TermIterator](#)  
*An iterator pointing to items in a list of terms.*
- class [Utf8Iterator](#)  
*An iterator which returns [Unicode](#) character values from a UTF-8 encoded string.*
- class [ValueIterator](#)  
*An iterator pointing to values associated with a document.*

## Typedefs

- typedef unsigned [doccount](#)  
*A count of documents.*
- typedef int [doccount\\_diff](#)  
*A signed difference between two counts of documents.*
- typedef unsigned [docid](#)  
*A unique identifier for a document.*
- typedef double [doclength](#)  
*A normalised document length.*
- typedef int [percent](#)  
*The percentage score for a document in an [MSet](#).*

- typedef unsigned [termcount](#)  
*A counts of terms.*
- typedef int [termcount\\_diff](#)  
*A signed difference between two counts of terms.*
- typedef unsigned [termpos](#)  
*A term position within a document or query.*
- typedef int [termpos\\_diff](#)  
*A signed difference between two term positions.*
- typedef unsigned [timeout](#)  
*A timeout value in milliseconds.*
- typedef unsigned [valueno](#)  
*The number for a value slot in a document.*
- typedef int [valueno\\_diff](#)  
*A signed difference between two value slot numbers.*
- typedef double [weight](#)  
*The weight of a document or term.*

## Functions

- bool [operator==](#) (const [MSetIterator](#) &a, const [MSetIterator](#) &b)  
*Equality test for [MSetIterator](#) objects.*
- bool [operator!=](#) (const [MSetIterator](#) &a, const [MSetIterator](#) &b)  
*Inequality test for [MSetIterator](#) objects.*
- bool [operator==](#) (const [ESetIterator](#) &a, const [ESetIterator](#) &b)  
*Equality test for [ESetIterator](#) objects.*
- bool [operator!=](#) (const [ESetIterator](#) &a, const [ESetIterator](#) &b)  
*Inequality test for [ESetIterator](#) objects.*
- bool [operator==](#) (const [PositionIterator](#) &a, const [PositionIterator](#) &b)  
*Test equality of two [PositionIterators](#).*
- bool [operator!=](#) (const [PositionIterator](#) &a, const [PositionIterator](#) &b)  
*Test inequality of two [PositionIterators](#).*
- bool [operator==](#) (const [PostingIterator](#) &a, const [PostingIterator](#) &b)

*Test equality of two PostingIterators.*

- bool `operator!=` (const [PostingIterator](#) &a, const [PostingIterator](#) &b)

*Test inequality of two PostingIterators.*

- std::string `sortable_serialise` (double value)

*Convert a floating point number to a string, preserving sort order.*

- double `sortable_unserialise` (const std::string &value)

*Convert a string encoded using sortable\_serialise back to a floating point number.*

- bool `operator==` (const [TermIterator](#) &a, const [TermIterator](#) &b)

*Equality test for [TermIterator](#) objects.*

- bool `operator!=` (const [TermIterator](#) &a, const [TermIterator](#) &b)

*Inequality test for [TermIterator](#) objects.*

- bool `operator==` (const [ValueIterator](#) &a, const [ValueIterator](#) &b)

*Equality test for [ValueIterator](#) objects.*

- bool `operator!=` (const [ValueIterator](#) &a, const [ValueIterator](#) &b)

*Inequality test for [ValueIterator](#) objects.*

- const char \* `version_string` ()

*Report the version string of the library which the program is linked with.*

- const char \* `xapian_version_string` ()

*For compatibility with [Xapian](#) 0.9.5 and earlier.*

- int `major_version` ()

*Report the major version of the library which the program is linked with.*

- int `xapian_major_version` ()

*For compatibility with [Xapian](#) 0.9.5 and earlier.*

- int `minor_version` ()

*Report the minor version of the library which the program is linked with.*

- int `xapian_minor_version` ()

*For compatibility with [Xapian](#) 0.9.5 and earlier.*

- int `revision` ()

*Report the revision of the library which the program is linked with.*

- int `xapian_revision` ()

*For compatibility with [Xapian](#) 0.9.5 and earlier.*

## Variables

- const int [DB\\_CREATE\\_OR\\_OPEN](#) = 1  
*Open for read/write; create if no db exists.*
- const int [DB\\_CREATE](#) = 2  
*Create a new database; fail if db exists.*
- const int [DB\\_CREATE\\_OR\\_OVERWRITE](#) = 3  
*Overwrite existing db; create if none exists.*
- const int [DB\\_OPEN](#) = 4  
*Open for read/write; fail if no db exists.*
- const [valueno](#) [BAD\\_VALUENO](#) = static\_cast<[valueno](#)>(-1)  
*Reserved value to indicate "no valueno".*

### 6.1.1 Detailed Description

The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 typedef unsigned Xapian::doccount

A count of documents.

This is used to hold values such as the number of documents in a database and the frequency of a term in the database.

#### 6.1.2.2 typedef int Xapian::doccount\_diff

A signed difference between two counts of documents.

This is used by the [Xapian](#) classes which are STL containers of documents for "difference\_type".

#### 6.1.2.3 typedef unsigned Xapian::docid

A unique identifier for a document.

Docid 0 is invalid, providing an "out of range" value which can be used to mean "not a valid document".



**6.1.2.4 typedef double Xapian::doclength**

A normalised document length.

The normalised document length is the document length divided by the average document length in the database.

**6.1.2.5 typedef int Xapian::percent**

The percentage score for a document in an [MSet](#).

**6.1.2.6 typedef unsigned Xapian::termcount**

A counts of terms.

This is used to hold values such as the Within [Document](#) Frequency (wdf).

**6.1.2.7 typedef int Xapian::termcount\_diff**

A signed difference between two counts of terms.

This is used by the [Xapian](#) classes which are STL containers of terms for "difference\_type".

**6.1.2.8 typedef unsigned Xapian::termpos**

A term position within a document or query.

**6.1.2.9 typedef int Xapian::termpos\_diff**

A signed difference between two term positions.

This is used by the [Xapian](#) classes which are STL containers of positions for "difference\_type".

**6.1.2.10 typedef unsigned Xapian::timeout**

A timeout value in milliseconds.

There are 1000 milliseconds in a second, so for example, to set a timeout of 5 seconds use 5000.

**6.1.2.11 typedef unsigned Xapian::valueno**

The number for a value slot in a document.

Any value slot number except [Xapian::BAD\\_VALUENO](#) is valid.

### 6.1.2.12 `typedef int Xapian::valueno_diff`

A signed difference between two value slot numbers.

This is used by the [Xapian](#) classes which are STL containers of values for "difference\_type".

### 6.1.2.13 `typedef double Xapian::weight`

The weight of a document or term.

## 6.1.3 Function Documentation

### 6.1.3.1 `int Xapian::major_version ()`

Report the major version of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN\_MAJOR\_VERSION) if shared libraries are being used.

### 6.1.3.2 `int Xapian::minor_version ()`

Report the minor version of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN\_MINOR\_VERSION) if shared libraries are being used.

### 6.1.3.3 `bool Xapian::operator!= (const ValueIterator & a, const ValueIterator & b) [inline]`

Inequality test for [ValueIterator](#) objects.

### 6.1.3.4 `bool Xapian::operator!= (const TermIterator & a, const TermIterator & b) [inline]`

Inequality test for [TermIterator](#) objects.

### 6.1.3.5 `bool Xapian::operator!= (const PostingIterator & a, const PostingIterator & b) [inline]`

Test inequality of two PostingIterators.

### 6.1.3.6 `bool Xapian::operator!= (const PositionIterator & a, const PositionIterator & b) [inline]`

Test inequality of two PositionIterators.

**6.1.3.7** `bool Xapian::operator!=(const ESetIterator & a, const ESetIterator & b)` `[inline]`

Inequality test for [ESetIterator](#) objects.

**6.1.3.8** `bool Xapian::operator!=(const MSetIterator & a, const MSetIterator & b)` `[inline]`

Inequality test for [MSetIterator](#) objects.

**6.1.3.9** `bool Xapian::operator==(const ValueIterator & a, const ValueIterator & b)` `[inline]`

Equality test for [ValueIterator](#) objects.

**6.1.3.10** `bool Xapian::operator==(const TermIterator & a, const TermIterator & b)` `[inline]`

Equality test for [TermIterator](#) objects.

**6.1.3.11** `bool Xapian::operator==(const PostingIterator & a, const PostingIterator & b)` `[inline]`

Test equality of two PostingIterators.

**6.1.3.12** `bool Xapian::operator==(const PositionIterator & a, const PositionIterator & b)` `[inline]`

Test equality of two PositionIterators.

**6.1.3.13** `bool Xapian::operator==(const ESetIterator & a, const ESetIterator & b)` `[inline]`

Equality test for [ESetIterator](#) objects.

**6.1.3.14** `bool Xapian::operator==(const MSetIterator & a, const MSetIterator & b)` `[inline]`

Equality test for [MSetIterator](#) objects.

**6.1.3.15** `int Xapian::revision()`

Report the revision of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN\_REVISION) if shared libraries are being used.

#### 6.1.3.16 `std::string Xapian::sortable_serialise (double value)`

Convert a floating point number to a string, preserving sort order.

This method converts a floating point number to a string, suitable for using as a value for numeric range restriction, or for use as a sort key.

The conversion is platform independent.

The conversion attempts to ensure that, for any pair of values supplied to the conversion algorithm, the result of comparing the original values (with a numeric comparison operator) will be the same as the result of comparing the resulting values (with a string comparison operator). On platforms which represent doubles with the precisions specified by IEEE\_754, this will be the case: if the representation of doubles is more precise, it is possible that two very close doubles will be mapped to the same string, so will compare equal.

Note also that both zero and -zero will be converted to the same representation: since these compare equal, this satisfies the comparison constraint, but it's worth knowing this if you wish to use the encoding in some situation where this distinction matters.

Handling of NaN isn't (currently) guaranteed to be sensible.

#### 6.1.3.17 `double Xapian::sortable_unserialise (const std::string & value)`

Convert a string encoded using *sortable\_serialise* back to a floating point number.

This expects the input to be a string produced by *sortable\_serialise()*. If the input is not such a string, the value returned is undefined (but no error will be thrown).

The result of the conversion will be exactly the value which was supplied to *sortable\_serialise()* when making the string on platforms which represent doubles with the precisions specified by IEEE\_754, but may be a different (nearby) value on other platforms.

#### 6.1.3.18 `const char* Xapian::version_string ()`

Report the version string of the library which the program is linked with.

This may be different to the version compiled against (given by XAPIAN\_VERSION) if shared libraries are being used.

#### 6.1.3.19 `int Xapian::xapian_major_version ()`

For compatibility with [Xapian 0.9.5](#) and earlier.

#### Deprecated

This function is now deprecated, use [Xapian::major\\_version\(\)](#) instead.

**6.1.3.20 int Xapian::xapian\_minor\_version ()**

For compatibility with [Xapian](#) 0.9.5 and earlier.

**Deprecated**

This function is now deprecated, use [Xapian::minor\\_version\(\)](#) instead.

**6.1.3.21 int Xapian::xapian\_revision ()**

For compatibility with [Xapian](#) 0.9.5 and earlier.

**Deprecated**

This function is now deprecated, use [Xapian::revision\(\)](#) instead.

**6.1.3.22 const char\* Xapian::xapian\_version\_string ()**

For compatibility with [Xapian](#) 0.9.5 and earlier.

**Deprecated**

This function is now deprecated, use [Xapian::version\\_string\(\)](#) instead.

**6.1.4 Variable Documentation****6.1.4.1 const valueno Xapian::BAD\_VALUENO = static\_cast<valueno>(-1)**

Reserved value to indicate "no valueno".

**6.1.4.2 const int Xapian::DB\_CREATE = 2**

Create a new database; fail if db exists.

**6.1.4.3 const int Xapian::DB\_CREATE\_OR\_OPEN = 1**

Open for read/write; create if no db exists.

**6.1.4.4 const int Xapian::DB\_CREATE\_OR\_OVERWRITE = 3**

Overwrite existing db; create if none exists.

**6.1.4.5 const int Xapian::DB\_OPEN = 4**

Open for read/write; fail if no db exists.

## 6.2 Xapian::Auto Namespace Reference

[Database](#) factory functions which determine the database type automatically.

### Functions

- [Database open\\_stub](#) (const std::string &file)  
*Construct a [Database](#) object for a stub database file.*

### 6.2.1 Detailed Description

[Database](#) factory functions which determine the database type automatically.

### 6.2.2 Function Documentation

#### 6.2.2.1 Database Xapian::Auto::open\_stub (const std::string &file)

Construct a [Database](#) object for a stub database file.

The stub database file contains serialised parameters for one or more databases.

#### Parameters:

*file* pathname of the stub database file.

## 6.3 Xapian::Flint Namespace Reference

[Database](#) factory functions for the flint backend.

### Functions

- [Database open](#) (const std::string &dir)  
*Construct a [Database](#) object for read-only access to a [Flint](#) database.*
- [WritableDatabase open](#) (const std::string &dir, int action, int block\_size=8192)  
*Construct a [Database](#) object for update access to a [Flint](#) database.*

### 6.3.1 Detailed Description

[Database](#) factory functions for the flint backend.

### 6.3.2 Function Documentation

#### 6.3.2.1 WritableDatabase Xapian::Flint::open (const std::string &dir, int action, int block\_size = 8192)

Construct a [Database](#) object for update access to a [Flint](#) database.

##### Parameters:

**dir** pathname of the directory containing the database.

**action** determines handling of existing/non-existing database:

- [Xapian::DB\\_CREATE](#) fail if database already exist, otherwise create new database.
- [Xapian::DB\\_CREATE\\_OR\\_OPEN](#) open existing database, or create new database if none exists.
- [Xapian::DB\\_CREATE\\_OR\\_OVERWRITE](#) overwrite existing database, or create new database if none exists.
- [Xapian::DB\\_OPEN](#) open existing database, failing if none exists.

**block\_size** the Btree blocksize to use (in bytes), which must be a power of two between 2048 and 65536 (inclusive). The default (also used if an invalid value is passed) is 8192 bytes. This parameter is ignored when opening an existing database.

#### 6.3.2.2 Database Xapian::Flint::open (const std::string &dir)

Construct a [Database](#) object for read-only access to a [Flint](#) database.

**Parameters:**

*dir* pathname of the directory containing the database.



## 6.4 Xapian::InMemory Namespace Reference

[Database](#) factory functions for the inmemory backend.

### Functions

- [WritableDatabase](#) `open ()`

*Construct a [WritableDatabase](#) object for a new, empty [InMemory](#) database.*

### 6.4.1 Detailed Description

[Database](#) factory functions for the inmemory backend.

### 6.4.2 Function Documentation

#### 6.4.2.1 [WritableDatabase](#) `Xapian::InMemory::open ()`

Construct a [WritableDatabase](#) object for a new, empty [InMemory](#) database.

Only a writable [InMemory](#) database can be created, since a read-only one would always remain empty.

## 6.5 Xapian::Quartz Namespace Reference

[Database](#) factory functions for the quartz backend.

### Functions

- [Database open](#) (const std::string &dir)  
*Construct a [Database](#) object for read-only access to a [Quartz](#) database.*
- [WritableDatabaseopen](#) (const std::string &dir, int action, int block\_size=8192)  
*Construct a [Database](#) object for update access to a [Quartz](#) database.*

### 6.5.1 Detailed Description

[Database](#) factory functions for the quartz backend.

### 6.5.2 Function Documentation

#### 6.5.2.1 Database Xapian::Quartz::open (const std::string & dir)

Construct a [Database](#) object for read-only access to a [Quartz](#) database.

The [Quartz](#) backend is deprecated - use the [Flint](#) backend instead.

##### Parameters:

*dir* pathname of the directory containing the database.

#### 6.5.2.2 Xapian::Quartz::WritableDatabaseopen (const std::string & dir, int action, int block\_size = 8192)

Construct a [Database](#) object for update access to a [Quartz](#) database.

The [Quartz](#) backend is deprecated - use the [Flint](#) backend instead.

##### Parameters:

*dir* pathname of the directory containing the database.

*action* determines handling of existing/non-existing database:

- [Xapian::DB\\_CREATE](#) fail if database already exist, otherwise create new database.
- [Xapian::DB\\_CREATE\\_OR\\_OPEN](#) open existing database, or create new database if none exists.
- [Xapian::DB\\_CREATE\\_OR\\_OVERWRITE](#) overwrite existing database, or create new database if none exists.

- [Xapian::DB\\_OPEN](#) open existing database, failing if none exists.

*block\_size* the Btree blocksize to use (in bytes), which must be a power of two between 2048 and 65536 (inclusive). The default (also used if an invalid value is passed) is 8192 bytes. This parameter is ignored when opening an existing database.

## 6.6 Xapian::Remote Namespace Reference

[Database](#) factory functions for the remote backend.

### Functions

- [Database open](#) (const std::string &host, unsigned int port, [Xapian::timeout timeout=10000](#), [Xapian::timeout connect\\_timeout=10000](#))  
Construct a [Database](#) object for read-only access to a remote database accessed via a TCP connection.
- [WritableDatabase open\\_writable](#) (const std::string &host, unsigned int port, [Xapian::timeout timeout=0](#), [Xapian::timeout connect\\_timeout=10000](#))  
Construct a [WritableDatabase](#) object for update access to a remote database accessed via a TCP connection.
- [Database open](#) (const std::string &program, const std::string &args, [Xapian::timeout timeout=10000](#))  
Construct a [Database](#) object for read-only access to a remote database accessed via a program.
- [WritableDatabase open\\_writable](#) (const std::string &program, const std::string &args, [Xapian::timeout timeout=0](#))  
Construct a [WritableDatabase](#) object for update access to a remote database accessed via a program.

### 6.6.1 Detailed Description

[Database](#) factory functions for the remote backend.

### 6.6.2 Function Documentation

#### 6.6.2.1 Database [Xapian::Remote::open](#) (const std::string & *program*, const std::string & *args*, [Xapian::timeout timeout](#) = 10000)

Construct a [Database](#) object for read-only access to a remote database accessed via a program.

Access to the remote database is done by running an external program and communicating with it on stdin/stdout.

#### Parameters:

*program* the external program to run.

*args* space-separated list of arguments to pass to program.

*timeout* timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then [Xapian::NetworkTimeoutError](#) is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

#### 6.6.2.2 Database `Xapian::Remote::open` (`const std::string & host`, `unsigned int port`, `Xapian::timeout timeout = 10000`, `Xapian::timeout connect_timeout = 10000`)

Construct a [Database](#) object for read-only access to a remote database accessed via a TCP connection.

Access to the remote database is via a TCP connection to the specified host and port.

##### Parameters:

*host* hostname to connect to.

*port* port number to connect to.

*timeout* timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then [Xapian::NetworkTimeoutError](#) is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

*connect\_timeout* timeout to use when connecting to the server. If this timeout is exceeded then [Xapian::NetworkTimeoutError](#) is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

#### 6.6.2.3 WritableDatabase `Xapian::Remote::open_writable` (`const std::string & program`, `const std::string & args`, `Xapian::timeout timeout = 0`)

Construct a [WritableDatabase](#) object for update access to a remote database accessed via a program.

Access to the remote database is done by running an external program and communicating with it on stdin/stdout.

##### Parameters:

*program* the external program to run.

*args* space-separated list of arguments to pass to program.

*timeout* timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then [Xapian::NetworkTimeoutError](#) is thrown. (Default is 0, which means don't timeout).

#### 6.6.2.4 WritableDatabase Xapian::Remote::open\_writable (const std::string & *host*, unsigned int *port*, Xapian::timeout *timeout* = 0, Xapian::timeout *connect\_timeout* = 10000)

Construct a [WritableDatabase](#) object for update access to a remote database accessed via a TCP connection.

Access to the remote database is via a TCP connection to the specified host and port.

##### Parameters:

*host* hostname to connect to.

*port* port number to connect to.

*timeout* timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then [Xapian::NetworkTimeoutError](#) is thrown. (Default is 0, which means don't timeout).

*connect\_timeout* timeout to use when connecting to the server. If this timeout is exceeded then [Xapian::NetworkTimeoutError](#) is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).

## 6.7 Xapian::Unicode Namespace Reference

Functions associated with handling [Unicode](#) characters.

### Enumerations

- enum [category](#) {  
UNASSIGNED,   UPPERCASE\_LETTER,   LOWERCASE\_LETTER,  
TITLECASE\_LETTER,  
MODIFIER\_LETTER,   OTHER\_LETTER,   NON\_SPACING\_MARK,  
ENCLOSING\_MARK,  
COMBINING\_SPACING\_MARK,       DECIMAL\_DIGIT\_NUMBER,  
LETTER\_NUMBER, OTHER\_NUMBER,  
SPACE\_SEPARATOR,       LINE\_SEPARATOR,       PARAGRAPH\_  
SEPARATOR, CONTROL,  
FORMAT,       PRIVATE\_USE,       SURROGATE,       CONNECTOR\_  
PUNCTUATION,  
DASH\_PUNCTUATION,       OPEN\_PUNCTUATION,       CLOSE\_  
PUNCTUATION, INITIAL\_QUOTE\_PUNCTUATION,  
FINAL\_QUOTE\_PUNCTUATION, OTHER\_PUNCTUATION, MATH\_  
SYMBOL, CURRENCY\_SYMBOL,  
MODIFIER\_SYMBOL, OTHER\_SYMBOL }

*Each Unicode character is in exactly one of these categories.*

### Functions

- unsigned [nonascii\\_to\\_utf8](#) (unsigned ch, char \*buf)  
*Convert a single non-ASCII [Unicode](#) character to UTF-8.*
- unsigned [to\\_utf8](#) (unsigned ch, char \*buf)  
*Convert a single [Unicode](#) character to UTF-8.*
- void [append\\_utf8](#) (std::string &s, unsigned ch)  
*Append the UTF-8 representation of a single [Unicode](#) character to a std::string.*
- [category get\\_category](#) (unsigned ch)  
*Return the category which a given [Unicode](#) character falls into.*
- bool [is\\_wordchar](#) (unsigned ch)  
*Test if a given [Unicode](#) character is "word character".*
- bool [is\\_whitespace](#) (unsigned ch)  
*Test if a given [Unicode](#) character is a whitespace character.*

- bool `is_currency` (unsigned *ch*)  
*Test if a given [Unicode](#) character is a currency symbol.*
- unsigned `tolower` (unsigned *ch*)  
*Convert a [Unicode](#) character to lowercase.*
- unsigned `toupper` (unsigned *ch*)  
*Convert a [Unicode](#) character to uppercase.*
- `std::string tolower` (const `std::string` &*term*)  
*Convert a UTF-8 `std::string` to lowercase.*
- `std::string toupper` (const `std::string` &*term*)  
*Convert a UTF-8 `std::string` to uppercase.*

### 6.7.1 Detailed Description

Functions associated with handling [Unicode](#) characters.

### 6.7.2 Enumeration Type Documentation

#### 6.7.2.1 enum `Xapian::Unicode::category`

Each [Unicode](#) character is in exactly one of these categories.

### 6.7.3 Function Documentation

#### 6.7.3.1 void `Xapian::Unicode::append_utf8` (`std::string` & *s*, unsigned *ch*) [inline]

Append the UTF-8 representation of a single [Unicode](#) character to a `std::string`.

References to `_utf8()`.

Referenced by `tolower()`, and `toupper()`.

#### 6.7.3.2 category `Xapian::Unicode::get_category` (unsigned *ch*) [inline]

Return the category which a given [Unicode](#) character falls into.

Referenced by `is_currency()`, `is_whitespace()`, and `is_wordchar()`.



**6.7.3.3 bool Xapian::Unicode::is\_currency (unsigned *ch*) [inline]**

Test if a given [Unicode](#) character is a currency symbol.

References `get_category()`.

**6.7.3.4 bool Xapian::Unicode::is\_whitespace (unsigned *ch*) [inline]**

Test if a given [Unicode](#) character is a whitespace character.

References `get_category()`.

**6.7.3.5 bool Xapian::Unicode::is\_wordchar (unsigned *ch*) [inline]**

Test if a given [Unicode](#) character is "word character".

References `get_category()`.

**6.7.3.6 unsigned Xapian::Unicode::nonascii\_to\_utf8 (unsigned *ch*, char \* *buf*)**

Convert a single non-ASCII [Unicode](#) character to UTF-8.

This is intended mainly as a helper method for `to_utf8()`.

The character *ch* (which must be  $> 128$ ) is written to the buffer *buf* and the length of the resultant UTF-8 character is returned.

NB *buf* must have space for (at least) 4 bytes.

Referenced by `to_utf8()`.

**6.7.3.7 unsigned Xapian::Unicode::to\_utf8 (unsigned *ch*, char \* *buf*)  
[inline]**

Convert a single [Unicode](#) character to UTF-8.

The character *ch* is written to the buffer *buf* and the length of the resultant UTF-8 character is returned.

NB *buf* must have space for (at least) 4 bytes.

References `nonascii_to_utf8()`.

Referenced by `append_utf8()`.

**6.7.3.8 std::string Xapian::Unicode::tolower (const std::string & *term*)  
[inline]**

Convert a UTF-8 `std::string` to lowercase.

References `append_utf8()`, and `tolower()`.

**6.7.3.9 unsigned Xapian::Unicode::tolower (unsigned *ch*)** `[inline]`

Convert a [Unicode](#) character to lowercase.

Referenced by `tolower()`.

**6.7.3.10 std::string Xapian::Unicode::toupper (const std::string & *term*)**  
`[inline]`

Convert a UTF-8 `std::string` to uppercase.

References `append_utf8()`, and `toupper()`.

**6.7.3.11 unsigned Xapian::Unicode::toupper (unsigned *ch*)** `[inline]`

Convert a [Unicode](#) character to uppercase.

Referenced by `toupper()`.

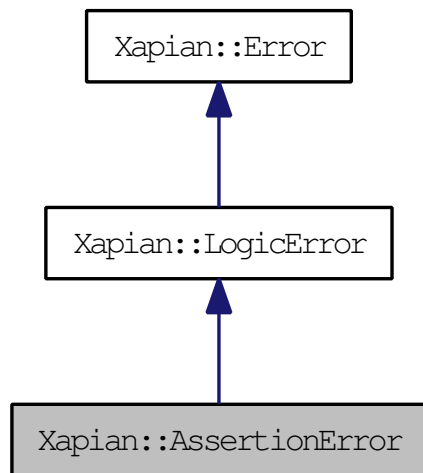
## Chapter 7

# Class Documentation

### 7.1 Xapian::AssertionError Class Reference

[AssertionError](#) is thrown if a logical assertion inside [Xapian](#) fails.

Inheritance diagram for Xapian::AssertionError:



#### Public Member Functions

- [AssertionError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)  
*General purpose constructor which allows setting errno.*
- [AssertionError](#) (const std::string &msg\_, int errno\_)  
*Construct from message and errno value.*

### 7.1.1 Detailed Description

[AssertionError](#) is thrown if a logical assertion inside [Xapian](#) fails.

In a debug build of [Xapian](#), a failed assertion in the core library code will cause [AssertionError](#) to be thrown.

This represents a bug in [Xapian](#) (either an invariant, precondition, etc has been violated, or the assertion is incorrect!)

### 7.1.2 Constructor & Destructor Documentation

**7.1.2.1 [Xapian::AssertionError::AssertionError](#) (const std::string & *msg\_*, const std::string & *context\_* = "", int *errno\_* = 0) [inline, explicit]**

General purpose constructor which allows setting errno.

**7.1.2.2 [Xapian::AssertionError::AssertionError](#) (const std::string & *msg\_*, int *errno\_*) [inline]**

Construct from message and errno value.

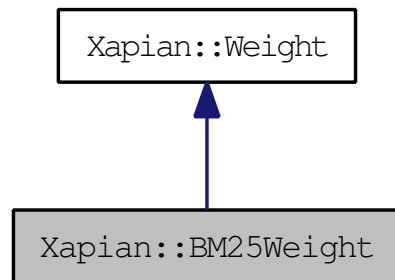
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.2 Xapian::BM25Weight Class Reference

BM25 weighting scheme.

Inheritance diagram for Xapian::BM25Weight:



### Public Member Functions

- [BM25Weight](#) (double k1\_, double k2\_, double k3\_, double b\_, double min\_normlen\_)  
*Construct a BM25 weight.*
- [BM25Weight \\* clone](#) () const  
*Return a new weight object of this type.*
- std::string [name](#) () const  
*Name of the weighting scheme.*
- std::string [serialise](#) () const  
*Serialise object parameters into a string.*
- [BM25Weight \\* unserialise](#) (const std::string &s) const  
*Create object given string serialisation returned by [serialise\(\)](#).*
- [Xapian::weight get\\_sumpart](#) ([Xapian::termcount](#) wdf, [Xapian::doclength](#) len) const  
*Get a weight which is part of the sum over terms being performed.*
- [Xapian::weight get\\_maxpart](#) () const  
*Gets the maximum value that [get\\_sumpart\(\)](#) may return.*
- [Xapian::weight get\\_sumextra](#) ([Xapian::doclength](#) len) const  
*Get an extra weight for a document to add to the sum calculated over the query terms.*
- [Xapian::weight get\\_maxextra](#) () const

*Gets the maximum value that `get_sumextra()` may return.*

- bool `get_sumpart_needs_doclength()` const  
*return false if the weight object doesn't need doclength*

### 7.2.1 Detailed Description

BM25 weighting scheme.

BM25 weighting options : The BM25 formula is

$$\frac{k_2 \cdot n_q}{1 + L_d} + \sum_t \frac{(k_3 + 1)q_t}{k_3 + q_t} \cdot \frac{(k_1 + 1)f_{t,d}}{k_1((1 - b) + bL_d) + f_{t,d}} \cdot w_t$$

where

- $w_t$  is the termweight of term  $t$
- $f_{t,d}$  is the within document frequency of term  $t$  in document  $d$
- $q_t$  is the within query frequency of term  $t$
- $L_d$  is the normalised length of document  $d$
- $n_q$  is the size of the query
- $k_1, k_2, k_3$  and  $b$  are user specified parameters

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 `Xapian::BM25Weight::BM25Weight (double $k1\_$ , double $k2\_$ , double $k3\_$ , double $b\_$ , double $min\_normlen\_$ )` [inline]

Construct a BM25 weight.

##### Parameters:

- $k1$**  governs the importance of within document frequency. Must be  $\geq 0$ . 0 means ignore wdf. Default is 1.
- $k2$**  compensation factor for the high wdf values in large documents. Must be  $\geq 0$ . 0 means no compensation. Default is 0.
- $k3$**  governs the importance of within query frequency. Must be  $\geq 0$ . 0 means ignore wqf. Default is 1.
- $b$**  Relative importance of within document frequency and document length. Must be  $\geq 0$  and  $\leq 1$ . Default is 0.5.
- $min\_normlen$**  specifies a cutoff on the minimum value that can be used for a normalised document length - smaller values will be forced up to this cutoff. This prevents very small documents getting a huge bonus weight. Default is 0.5.

### 7.2.3 Member Function Documentation

#### 7.2.3.1 BM25Weight\* Xapian::BM25Weight::clone () const [virtual]

Return a new weight object of this type.

A subclass called FooWeight taking parameters param1 and param2 should implement this as:

```
virtual FooWeight * clone() const { return new FooWeight(param1, param2); }
```

Implements [Xapian::Weight](#).

#### 7.2.3.2 Xapian::weight Xapian::BM25Weight::get\_maxextra () const [virtual]

Gets the maximum value that [get\\_sumextra\(\)](#) may return.

This is used in optimising searches.

Implements [Xapian::Weight](#).

#### 7.2.3.3 Xapian::weight Xapian::BM25Weight::get\_maxpart () const [virtual]

Gets the maximum value that [get\\_sumpart\(\)](#) may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implements [Xapian::Weight](#).

#### 7.2.3.4 Xapian::weight Xapian::BM25Weight::get\_sumextra (Xapian::doclength *len*) const [virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

##### Parameters:

*len* the (unnormalised) document length.

Implements [Xapian::Weight](#).

#### 7.2.3.5 Xapian::weight Xapian::BM25Weight::get\_sumpart (Xapian::termcount *wdf*, Xapian::doclength *len*) const [virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

**Parameters:**

*wdf* the within document frequency of the term.

*len* the (unnormalised) document length.

Implements [Xapian::Weight](#).

**7.2.3.6** `bool Xapian::BM25Weight::get_sumpart_needs_doclength () const` [virtual]

return false if the weight object doesn't need doclength

Reimplemented from [Xapian::Weight](#).

**7.2.3.7** `std::string Xapian::BM25Weight::name () const` [virtual]

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implements [Xapian::Weight](#).

**7.2.3.8** `std::string Xapian::BM25Weight::serialise () const` [virtual]

Serialise object parameters into a string.

Implements [Xapian::Weight](#).

**7.2.3.9** `BM25Weight* Xapian::BM25Weight::unserialise (const std::string & s) const` [virtual]

Create object given string serialisation returned by [serialise\(\)](#).

Implements [Xapian::Weight](#).

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

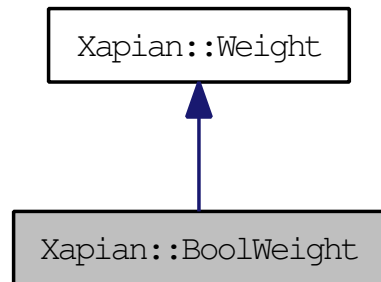
- [xapian/enquire.h](#)



## 7.3 Xapian::BoolWeight Class Reference

Boolean weighting scheme (everything gets 0).

Inheritance diagram for Xapian::BoolWeight:



### Public Member Functions

- `BoolWeight * clone () const`  
*Return a new weight object of this type.*
- `std::string name () const`  
*Name of the weighting scheme.*
- `std::string serialise () const`  
*Serialise object parameters into a string.*
- `BoolWeight * unserialise (const std::string &s) const`  
*Create object given string serialisation returned by `serialise()`.*
- `Xapian::weight get_sumpart (Xapian::termcount wdf, Xapian::doclength len) const`  
*Get a weight which is part of the sum over terms being performed.*
- `Xapian::weight get_maxpart () const`  
*Gets the maximum value that `get_sumpart()` may return.*
- `Xapian::weight get_sumextra (Xapian::doclength len) const`  
*Get an extra weight for a document to add to the sum calculated over the query terms.*
- `Xapian::weight get_maxextra () const`  
*Gets the maximum value that `get_sumextra()` may return.*
- `bool get_sumpart_needs_doclength () const`  
*return false if the weight object doesn't need doclength*

### 7.3.1 Detailed Description

Boolean weighting scheme (everything gets 0).

### 7.3.2 Member Function Documentation

#### 7.3.2.1 BoolWeight\* Xapian::BoolWeight::clone () const [virtual]

Return a new weight object of this type.

A subclass called FooWeight taking parameters param1 and param2 should implement this as:

```
virtual FooWeight * clone() const { return new FooWeight(param1, param2); }
```

Implements [Xapian::Weight](#).

#### 7.3.2.2 Xapian::weight Xapian::BoolWeight::get\_maxextra () const [virtual]

Gets the maximum value that [get\\_sumextra\(\)](#) may return.

This is used in optimising searches.

Implements [Xapian::Weight](#).

#### 7.3.2.3 Xapian::weight Xapian::BoolWeight::get\_maxpart () const [virtual]

Gets the maximum value that [get\\_sumpart\(\)](#) may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implements [Xapian::Weight](#).

#### 7.3.2.4 Xapian::weight Xapian::BoolWeight::get\_sumextra (Xapian::doclength len) const [virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

#### Parameters:

*len* the (unnormalised) document length.

Implements [Xapian::Weight](#).

### 7.3.2.5 Xapian::weight Xapian::BoolWeight::get\_sumpart (Xapian::termcount *wdf*, Xapian::doclength *len*) const [virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

#### Parameters:

*wdf* the within document frequency of the term.

*len* the (unnormalised) document length.

Implements [Xapian::Weight](#).

### 7.3.2.6 bool Xapian::BoolWeight::get\_sumpart\_needs\_doclength () const [virtual]

return false if the weight object doesn't need doclength

Reimplemented from [Xapian::Weight](#).

### 7.3.2.7 std::string Xapian::BoolWeight::name () const [virtual]

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implements [Xapian::Weight](#).

### 7.3.2.8 std::string Xapian::BoolWeight::serialise () const [virtual]

Serialise object parameters into a string.

Implements [Xapian::Weight](#).

### 7.3.2.9 BoolWeight\* Xapian::BoolWeight::unserialise (const std::string & *s*) const [virtual]

Create object given string serialisation returned by [serialise\(\)](#).

Implements [Xapian::Weight](#).

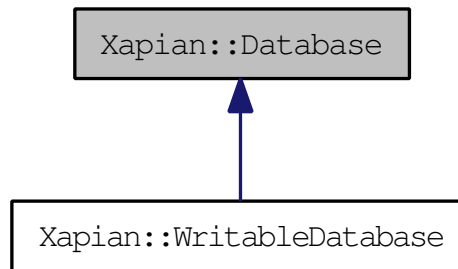
The documentation for this class was generated from the following file:

- [xapian/enquire.h](#)

## 7.4 Xapian::Database Class Reference

This class is used to access a database, or a group of databases.

Inheritance diagram for Xapian::Database:



### Public Member Functions

- void [add\\_database](#) (const [Database](#) &database)  
*Add an existing database (or group of databases) to those accessed by this object.*
- [Database](#) ()  
*Create a [Database](#) with no databases in.*
- [Database](#) (const std::string &path)  
*Open a [Database](#), automatically determining the database backend to use.*
- virtual [~Database](#) ()  
*Destroy this handle on the database.*
- [Database](#) (const [Database](#) &other)  
*Copying is allowed.*
- void [operator=](#) (const [Database](#) &other)  
*Assignment is allowed.*
- void [reopen](#) ()  
*Re-open the database.*
- virtual std::string [get\\_description](#) () const  
*Return a string describing this object.*
- [PostingIterator](#) [postlist\\_begin](#) (const std::string &name) const  
*An iterator pointing to the start of the postlist for a given term.*
- [PostingIterator](#) [postlist\\_end](#) (const std::string &) const

Corresponding end iterator to [postlist\\_begin\(\)](#).

- [TermIterator termlist\\_begin](#) (Xapian::docid did) const  
*An iterator pointing to the start of the termlist for a given document.*
- [TermIterator termlist\\_end](#) (Xapian::docid) const  
*Corresponding end iterator to [termlist\\_begin\(\)](#).*
- [bool has\\_positions](#) () const  
*Does this database have any positional information?*
- [PositionIterator positionlist\\_begin](#) (Xapian::docid did, const std::string &name) const  
*An iterator pointing to the start of the position list for a given term in a given document.*
- [PositionIterator positionlist\\_end](#) (Xapian::docid, const std::string &) const  
*Corresponding end iterator to [positionlist\\_begin\(\)](#).*
- [TermIterator allterms\\_begin](#) () const  
*An iterator which runs across all terms in the database.*
- [TermIterator allterms\\_end](#) () const  
*Corresponding end iterator to [allterms\\_begin\(\)](#).*
- [TermIterator allterms\\_begin](#) (const std::string &prefix) const  
*An iterator which runs across all terms with a given prefix.*
- [TermIterator allterms\\_end](#) (const std::string &) const  
*Corresponding end iterator to [allterms\\_begin\(prefix\)](#).*
- [Xapian::doccount get\\_doccount](#) () const  
*Get the number of documents in the database.*
- [Xapian::docid get\\_lastdocid](#) () const  
*Get the highest document id which has been used in the database.*
- [Xapian::doclength get\\_avlength](#) () const  
*Get the average length of the documents in the database.*
- [Xapian::doccount get\\_termfreq](#) (const std::string &name) const  
*Get the number of documents in the database indexed by a given term.*
- [bool term\\_exists](#) (const std::string &name) const  
*Check if a given term exists in the database.*

- [Xapian::termcount get\\_collection\\_freq](#) (const std::string &name) const  
*Return the total number of occurrences of the given term.*
- [Xapian::doclength get\\_doclength](#) (Xapian::docid did) const  
*Get the length of a document.*
- void [keep\\_alive](#) ()  
*Send a "keep-alive" to remote databases to stop them timing out.*
- [Xapian::Document get\\_document](#) (Xapian::docid did) const  
*Get a document from the database, given its document id.*
- std::string [get\\_spelling\\_suggestion](#) (const std::string &word, unsigned max\_edit\_distance=2) const  
*Suggest a spelling correction.*
- [Xapian::TermIterator spellings\\_begin](#) () const  
*An iterator which returns all the spelling correction targets.*
- [Xapian::TermIterator spellings\\_end](#) () const  
*Corresponding end iterator to [spellings\\_begin](#)().*
- [Xapian::TermIterator synonyms\\_begin](#) (const std::string &term) const  
*An iterator which returns all the synonyms for a given term.*
- [Xapian::TermIterator synonyms\\_end](#) (const std::string &) const  
*Corresponding end iterator to [synonyms\\_begin](#)(term).*
- [Xapian::TermIterator synonym\\_keys\\_begin](#) (const std::string &prefix="") const  
*An iterator which returns all terms which have synonyms.*
- [Xapian::TermIterator synonym\\_keys\\_end](#) (const std::string &="") const  
*Corresponding end iterator to [synonym\\_keys\\_begin](#)(prefix).*
- std::string [get\\_metadata](#) (const std::string &key) const  
*Get the user-specified metadata associated with a given key.*
- [Xapian::TermIterator metadata\\_keys\\_begin](#) (const std::string &prefix="") const  
*An iterator which returns all user-specified metadata keys.*
- [Xapian::TermIterator metadata\\_keys\\_end](#) (const std::string &="") const  
*Corresponding end iterator to [metadata\\_keys\\_begin](#)().*

### 7.4.1 Detailed Description

This class is used to access a database, or a group of databases.

For searching, this class is used in conjunction with an [Enquire](#) object.

#### Exceptions:

[\*InvalidArgumentError\*](#) will be thrown if an invalid argument is supplied, for example, an unknown database type.

[\*DatabaseOpeningError\*](#) may be thrown if the database cannot be opened (for example, a required file cannot be found).

[\*DatabaseVersionError\*](#) may be thrown if the database is in an unsupported format (for example, created by a newer version of [Xapian](#) which uses an incompatible format).

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 Xapian::Database::Database ()

Create a [Database](#) with no databases in.

#### 7.4.2.2 Xapian::Database::Database (const std::string & *path*) [explicit]

Open a [Database](#), automatically determining the database backend to use.

#### Parameters:

*path* directory that the database is stored in.

#### 7.4.2.3 virtual Xapian::Database::~~Database () [virtual]

Destroy this handle on the database.

If there are no copies of this object remaining, the database(s) will be closed.

#### 7.4.2.4 Xapian::Database::Database (const Database & *other*)

Copying is allowed.

The internals are reference counted, so copying is cheap.

### 7.4.3 Member Function Documentation

#### 7.4.3.1 void Xapian::Database::add\_database (const Database & *database*)

Add an existing database (or group of databases) to those accessed by this object.

**Parameters:**

*database* the database(s) to add.

**7.4.3.2 TermIterator Xpian::Database::allterms\_begin (const std::string & prefix) const**

An iterator which runs across all terms with a given prefix.

This is functionally similar to getting an iterator with [allterms\\_begin\(\)](#) and then calling `skip_to(prefix)` on that iterator to move to the start of the prefix, but is more convenient (because it detects the end of the prefixed terms), and may be more efficient than simply calling `skip_to()` after opening the iterator, particularly for remote databases.

**Parameters:**

*prefix* The prefix to restrict the returned terms to.

**7.4.3.3 TermIterator Xpian::Database::allterms\_begin () const**

An iterator which runs across all terms in the database.

**7.4.3.4 TermIterator Xpian::Database::allterms\_end (const std::string &) const [inline]**

Corresponding end iterator to `allterms_begin(prefix)`.

**7.4.3.5 TermIterator Xpian::Database::allterms\_end () const [inline]**

Corresponding end iterator to [allterms\\_begin\(\)](#).

**7.4.3.6 Xpian::doclength Xpian::Database::get\_avlength () const**

Get the average length of the documents in the database.

**7.4.3.7 Xpian::termcount Xpian::Database::get\_collection\_freq (const std::string & tname) const**

Return the total number of occurrences of the given term.

This is the sum of the number of occurrences of the term in each document it indexes: i.e., the sum of the within document frequencies of the term.

**Parameters:**

*tname* The term whose collection frequency is being requested.



**7.4.3.8 virtual std::string Xapian::Database::get\_description () const**  
[virtual]

Return a string describing this object.

Reimplemented in [Xapian::WritableDatabase](#).

**7.4.3.9 Xapian::doccount Xapian::Database::get\_doccount () const**

Get the number of documents in the database.

**7.4.3.10 Xapian::doclength Xapian::Database::get\_doclength (Xapian::docid *did*) const**

Get the length of a document.

**7.4.3.11 Xapian::Document Xapian::Database::get\_document (Xapian::docid *did*) const**

Get a document from the database, given its document id.

This method returns a [Xapian::Document](#) object which provides the information about a document.

**Parameters:**

*did* The document id of the document to retrieve.

**Returns:**

A [Xapian::Document](#) object containing the document data

**Exceptions:**

[Xapian::DocNotFoundError](#) The document specified could not be found in the database.

**7.4.3.12 Xapian::docid Xapian::Database::get\_lastdocid () const**

Get the highest document id which has been used in the database.

**7.4.3.13 std::string Xapian::Database::get\_metadata (const std::string & *key*) const**

Get the user-specified metadata associated with a given key.

User-specified metadata allows you to store arbitrary information in the form of (key,tag) pairs. See [WritableDatabase::set\\_metadata\(\)](#) for more information.

When invoked on a [Xapian::Database](#) object representing multiple databases, currently only the metadata for the first is considered but this behaviour may change in the future.

If there is no piece of metadata associated with the specified key, an empty string is returned (this applies even for backends which don't support metadata).

Empty keys are not valid, and specifying one will cause an exception.

**Parameters:**

*key* The key of the metadata item to access.

**Returns:**

The retrieved metadata item's value.

**Exceptions:**

[Xapian::InvalidArgumentError](#) will be thrown if the key supplied is empty.

**7.4.3.14** `std::string Xapian::Database::get_spelling_suggestion (const std::string & word, unsigned max_edit_distance = 2) const`

Suggest a spelling correction.

**Parameters:**

*word* The potentially misspelled word.

*max\_edit\_distance* Only consider words which are at most *max\_edit\_distance* edits from *word*. An edit is a character insertion, deletion, or the transposition of two adjacent characters (default is 2).

**7.4.3.15** `Xapian::doccount Xapian::Database::get_termfreq (const std::string & tname) const`

Get the number of documents in the database indexed by a given term.

**7.4.3.16** `bool Xapian::Database::has_positions () const`

Does this database have any positional information?

**7.4.3.17** `void Xapian::Database::keep_alive ()`

Send a "keep-alive" to remote databases to stop them timing out.

**7.4.3.18 Xapian::TermIterator Xapian::Database::metadata\_keys\_begin (const std::string & *prefix* = "") const**

An iterator which returns all user-specified metadata keys.

When invoked on a [Xapian::Database](#) object representing multiple databases, currently only the metadata for the first is considered but this behaviour may change in the future.

If the backend doesn't support metadata, then this method returns an iterator which compares equal to that returned by [metadata\\_keys\\_end\(\)](#).

**Parameters:**

*prefix* If non-empty, only keys with this prefix are returned.

**Exceptions:**

[Xapian::UnimplementedError](#) will be thrown if the backend implements user-specified metadata, but doesn't implement iterating its keys (currently this happens for the [InMemory](#) backend).

**7.4.3.19 Xapian::TermIterator Xapian::Database::metadata\_keys\_end (const std::string & = "") const [inline]**

Corresponding end iterator to [metadata\\_keys\\_begin\(\)](#).

**7.4.3.20 void Xapian::Database::operator= (const Database & *other*)**

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

**7.4.3.21 PositionIterator Xapian::Database::positionlist\_begin (Xapian::docid *did*, const std::string & *tname*) const**

An iterator pointing to the start of the position list for a given term in a given document.

**7.4.3.22 PositionIterator Xapian::Database::positionlist\_end (Xapian::docid, const std::string &) const [inline]**

Corresponding end iterator to [positionlist\\_begin\(\)](#).

**7.4.3.23 PostingIterator Xapian::Database::postlist\_begin (const std::string & *tname*) const**

An iterator pointing to the start of the postlist for a given term.

If the term name is the empty string, the iterator returned will list all the documents in the database. Such an iterator will always return a WDF value of 1, since there is no obvious meaning for this quantity in this case.

#### 7.4.3.24 PostingIterator Xapian::Database::postlist\_end (const std::string &) const [inline]

Corresponding end iterator to [postlist\\_begin\(\)](#).

#### 7.4.3.25 void Xapian::Database::reopen ()

Re-open the database.

This re-opens the database(s) to the latest available version(s). It can be used either to make sure the latest results are returned, or to recover from a [Xapian::DatabaseModifiedError](#).

#### 7.4.3.26 Xapian::TermIterator Xapian::Database::spellings\_begin () const

An iterator which returns all the spelling correction targets.

This returns all the words which are considered as targets for the spelling correction algorithm. The frequency of each word is available as the term frequency of each entry in the returned iterator.

#### 7.4.3.27 Xapian::TermIterator Xapian::Database::spellings\_end () const [inline]

Corresponding end iterator to [spellings\\_begin\(\)](#).

#### 7.4.3.28 Xapian::TermIterator Xapian::Database::synonym\_keys\_begin (const std::string & *prefix* = "") const

An iterator which returns all terms which have synonyms.

##### Parameters:

*prefix* If non-empty, only terms with this prefix are returned.

#### 7.4.3.29 Xapian::TermIterator Xapian::Database::synonym\_keys\_end (const std::string & = "") const [inline]

Corresponding end iterator to [synonym\\_keys\\_begin\(prefix\)](#).

**7.4.3.30 Xapian::TermIterator Xapian::Database::synonyms\_begin (const std::string & *term*) const**

An iterator which returns all the synonyms for a given term.

**Parameters:**

*term* The term to return synonyms for.

**7.4.3.31 Xapian::TermIterator Xapian::Database::synonyms\_end (const std::string &) const [inline]**

Corresponding end iterator to synonyms\_begin(term).

**7.4.3.32 bool Xapian::Database::term\_exists (const std::string & *tname*) const**

Check if a given term exists in the database.

Return true if and only if the term exists in the database. This is the same as (get\_termfreq(*tname*) != 0), but will often be more efficient.

**7.4.3.33 TermIterator Xapian::Database::termlist\_begin (Xapian::docid *did*) const**

An iterator pointing to the start of the termlist for a given document.

**7.4.3.34 TermIterator Xapian::Database::termlist\_end (Xapian::docid) const [inline]**

Corresponding end iterator to [termlist\\_begin\(\)](#).

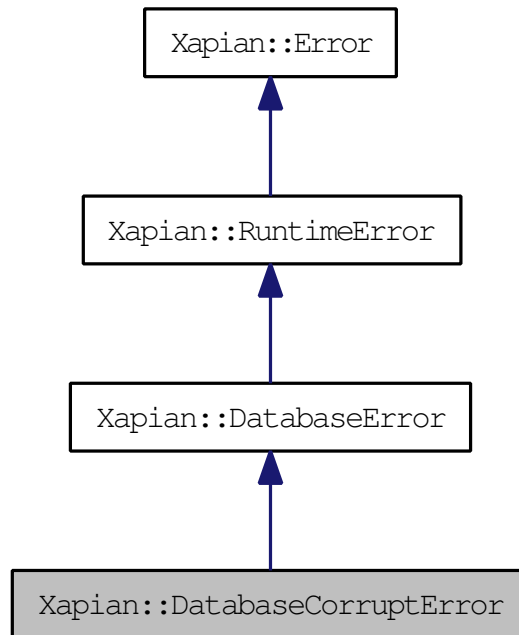
The documentation for this class was generated from the following file:

- [xapian/database.h](#)

## 7.5 Xapian::DatabaseCorruptError Class Reference

[DatabaseCorruptError](#) indicates database corruption was detected.

Inheritance diagram for Xapian::DatabaseCorruptError:



### Public Member Functions

- [DatabaseCorruptError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [DatabaseCorruptError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.5.1 Detailed Description

[DatabaseCorruptError](#) indicates database corruption was detected.

## 7.5.2 Constructor & Destructor Documentation

### 7.5.2.1 Xapian::DatabaseCorruptError::DatabaseCorruptError (const std::string & *msg\_*, const std::string & *context\_* = "", int *errno\_* = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.5.2.2 Xapian::DatabaseCorruptError::DatabaseCorruptError (const std::string & *msg\_*, int *errno\_*) [inline]

Construct from message and errno value.

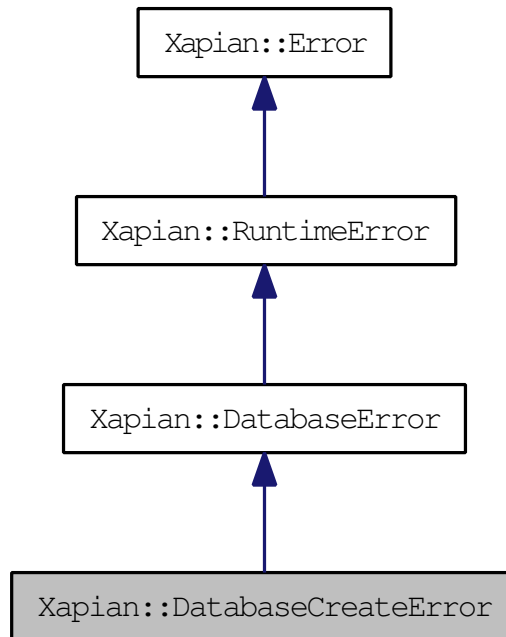
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.6 Xapian::DatabaseCreateError Class Reference

[DatabaseCreateError](#) indicates a failure to create a database.

Inheritance diagram for Xapian::DatabaseCreateError:



### Public Member Functions

- [DatabaseCreateError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [DatabaseCreateError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.6.1 Detailed Description

[DatabaseCreateError](#) indicates a failure to create a database.



## 7.6.2 Constructor & Destructor Documentation

### 7.6.2.1 Xapian::DatabaseCreateError::DatabaseCreateError (const std::string & msg\_, const std::string & context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.6.2.2 Xapian::DatabaseCreateError::DatabaseCreateError (const std::string & msg\_, int errno\_) [inline]

Construct from message and errno value.

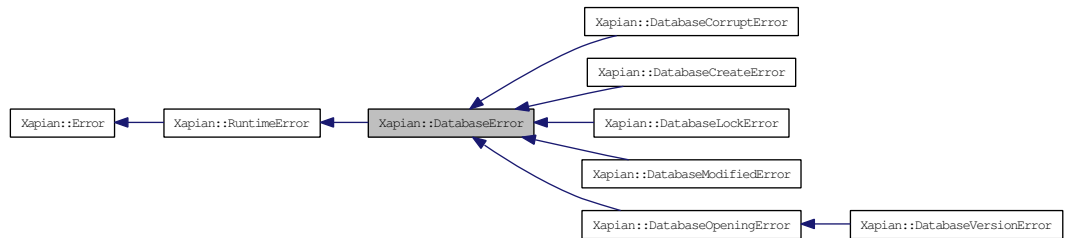
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.7 Xapian::DatabaseError Class Reference

[DatabaseError](#) indicates some sort of database related error.

Inheritance diagram for Xapian::DatabaseError:



### Public Member Functions

- [DatabaseError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)  
*General purpose constructor which allows setting errno.*
- [DatabaseError](#) (const std::string &msg\_, int errno\_)  
*Construct from message and errno value.*

### 7.7.1 Detailed Description

[DatabaseError](#) indicates some sort of database related error.

### 7.7.2 Constructor & Destructor Documentation

**7.7.2.1** `Xapian::DatabaseError::DatabaseError (const std::string & msg_, const std::string & context_ = "", int errno_ = 0) [inline, explicit]`

General purpose constructor which allows setting errno.

**7.7.2.2** `Xapian::DatabaseError::DatabaseError (const std::string & msg_, int errno_) [inline]`

Construct from message and errno value.

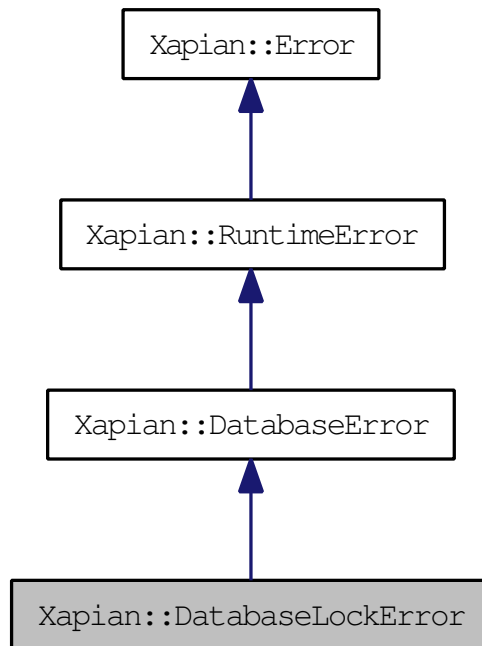
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.8 Xapian::DatabaseLockError Class Reference

[DatabaseLockError](#) indicates failure to lock a database.

Inheritance diagram for Xapian::DatabaseLockError:



### Public Member Functions

- [DatabaseLockError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [DatabaseLockError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.8.1 Detailed Description

[DatabaseLockError](#) indicates failure to lock a database.

## 7.8.2 Constructor & Destructor Documentation

### 7.8.2.1 `Xapian::DatabaseLockError::DatabaseLockError (const std::string & msg_, const std::string & context_ = "", int errno_ = 0) [inline, explicit]`

General purpose constructor which allows setting `errno`.

### 7.8.2.2 `Xapian::DatabaseLockError::DatabaseLockError (const std::string & msg_, int errno_)` [inline]

Construct from message and `errno` value.

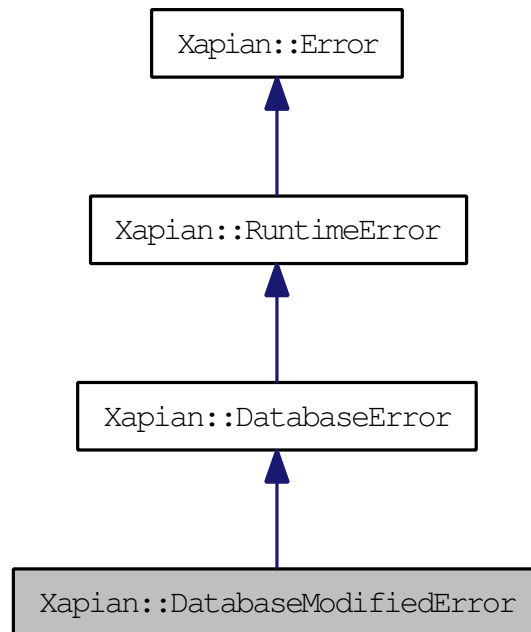
The documentation for this class was generated from the following file:

- `xapian/error.h`

## 7.9 Xapian::DatabaseModifiedError Class Reference

[DatabaseModifiedError](#) indicates a database was modified.

Inheritance diagram for Xapian::DatabaseModifiedError:



### Public Member Functions

- [DatabaseModifiedError](#) (const std::string &msg\_, const std::string &context\_ = "", int errno\_ = 0)

*General purpose constructor which allows setting errno.*

- [DatabaseModifiedError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.9.1 Detailed Description

[DatabaseModifiedError](#) indicates a database was modified.

To recover after catching this error, you need to call [Xapian::Database::reopen\(\)](#) on the [Database](#) and repeat the operation which failed.

## 7.9.2 Constructor & Destructor Documentation

### 7.9.2.1 `Xapian::DatabaseModifiedError::DatabaseModifiedError (const std::string & msg_, const std::string & context_ = "", int errno_ = 0) [inline, explicit]`

General purpose constructor which allows setting errno.

### 7.9.2.2 `Xapian::DatabaseModifiedError::DatabaseModifiedError (const std::string & msg_, int errno_) [inline]`

Construct from message and errno value.

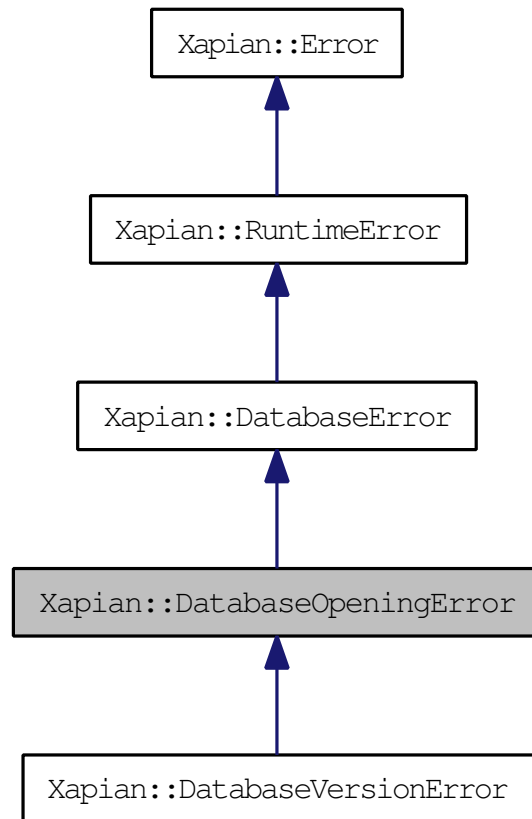
The documentation for this class was generated from the following file:

- `xapian/error.h`

## 7.10 Xapian::DatabaseOpeningError Class Reference

[DatabaseOpeningError](#) indicates failure to open a database.

Inheritance diagram for Xapian::DatabaseOpeningError:



### Public Member Functions

- [DatabaseOpeningError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)  
*General purpose constructor which allows setting errno.*
- [DatabaseOpeningError](#) (const std::string &msg\_, int errno\_)  
*Construct from message and errno value.*

### 7.10.1 Detailed Description

[DatabaseOpeningError](#) indicates failure to open a database.

## 7.10.2 Constructor & Destructor Documentation

### 7.10.2.1 Xapian::DatabaseOpeningError::DatabaseOpeningError (const std::string & *msg\_*, const std::string & *context\_* = "", int *errno\_* = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.10.2.2 Xapian::DatabaseOpeningError::DatabaseOpeningError (const std::string & *msg\_*, int *errno\_*) [inline]

Construct from message and errno value.

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

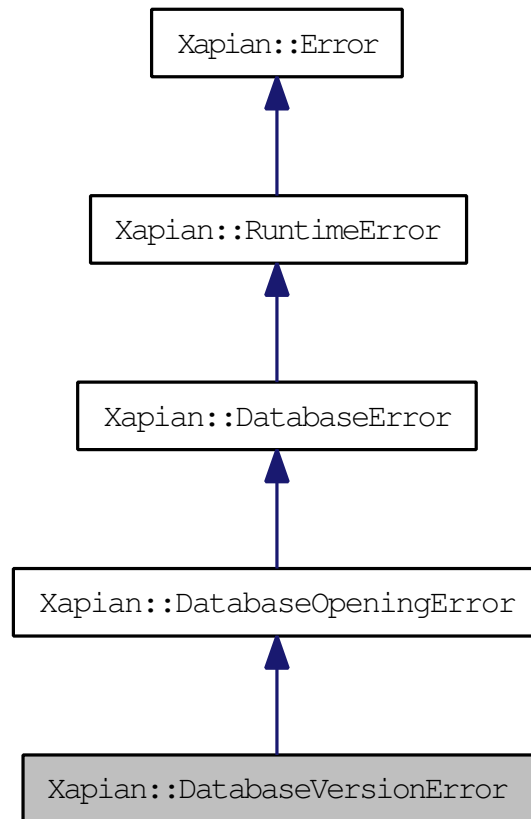
- xapian/[error.h](#)



## 7.11 Xapian::DatabaseVersionError Class Reference

[DatabaseVersionError](#) indicates that a database is in an unsupported format.

Inheritance diagram for Xapian::DatabaseVersionError:



### Public Member Functions

- [DatabaseVersionError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)  
*General purpose constructor which allows setting errno.*
- [DatabaseVersionError](#) (const std::string &msg\_, int errno\_)  
*Construct from message and errno value.*

#### 7.11.1 Detailed Description

[DatabaseVersionError](#) indicates that a database is in an unsupported format.

From time to time, new versions of [Xapian](#) will require the database format to be changed, to allow new information to be stored or new optimisations to be performed. Backwards compatibility will sometimes be maintained, so that new versions of [Xapian](#) can open old databases, but in some cases [Xapian](#) will be unable to open a database because it is in too old (or new) a format. This can be resolved either by upgrading or downgrading the version of [Xapian](#) in use, or by rebuilding the database from scratch with the current version of [Xapian](#).

## 7.11.2 Constructor & Destructor Documentation

### 7.11.2.1 `Xapian::DatabaseVersionError::DatabaseVersionError (const std::string & msg_, const std::string & context_ = "", int errno_ = 0) [inline, explicit]`

General purpose constructor which allows setting errno.

### 7.11.2.2 `Xapian::DatabaseVersionError::DatabaseVersionError (const std::string & msg_, int errno_) [inline]`

Construct from message and errno value.

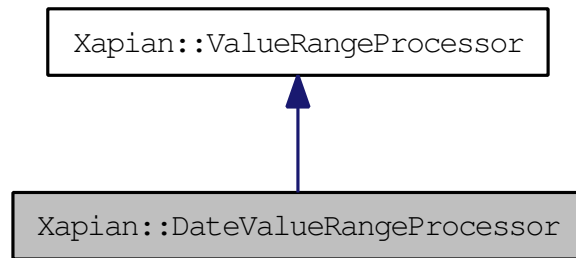
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.12 Xapian::DateValueRangeProcessor Class Reference

Handle a date range.

Inheritance diagram for Xapian::DateValueRangeProcessor:



### Public Member Functions

- [DateValueRangeProcessor](#) ([Xapian::valueno](#) valno\_, bool prefer\_mdy\_=false, int epoch\_year\_=1970)  
*Constructor.*
- [Xapian::valueno operator\(\)](#) (std::string &begin, std::string &end)  
*Check for a valid range of this type.*

#### 7.12.1 Detailed Description

Handle a date range.

Begin and end must be dates in a recognised format.

#### 7.12.2 Constructor & Destructor Documentation

##### 7.12.2.1 Xapian::DateValueRangeProcessor::DateValueRangeProcessor (Xapian::valueno valno\_, bool prefer\_mdy\_ = false, int epoch\_year\_ = 1970) [inline]

Constructor.

##### Parameters:

*valno\_* The value number to return from operator().

*prefer\_mdy\_* Should ambiguous dates be interpreted as month/day/year rather than day/month/year? (default: false)

*epoch\_year\_* Year to use as the epoch for dates with 2 digit years (default: 1970, so 1/1/69 is 2069 while 1/1/70 is 1970).

### 7.12.3 Member Function Documentation

#### 7.12.3.1 `Xapian::valueno Xapian::DateValueRangeProcessor::operator() (std::string & begin, std::string & end)` [virtual]

Check for a valid range of this type.

If BEGIN..END is a sensible date range, this method returns the value number of range filter on. Otherwise it returns [Xapian::BAD\\_VALUENO](#).

Implements [Xapian::ValueRangeProcessor](#).

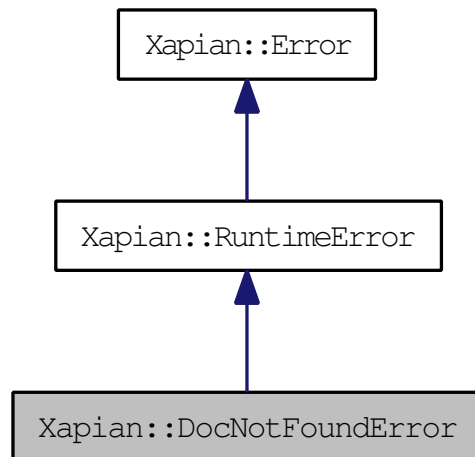
The documentation for this class was generated from the following file:

- [xapian/queryparser.h](#)

## 7.13 Xapian::DocNotFoundError Class Reference

Indicates an attempt to access a document not present in the database.

Inheritance diagram for Xapian::DocNotFoundError:



### Public Member Functions

- [DocNotFoundError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [DocNotFoundError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.13.1 Detailed Description

Indicates an attempt to access a document not present in the database.

### 7.13.2 Constructor & Destructor Documentation

#### 7.13.2.1 Xapian::DocNotFoundError::DocNotFoundError (const std::string &msg\_, const std::string &context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.

#### 7.13.2.2 Xapian::DocNotFoundError::DocNotFoundError (const std::string & msg\_, int *errno*\_) [inline]

Construct from message and errno value.

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

- xapian/[error.h](#)

## 7.14 Xapian::Document Class Reference

A handle representing a document in a [Xapian](#) database.

### Public Member Functions

- [Document](#) (const [Document](#) &other)  
*Copying is allowed.*
- void [operator=](#) (const [Document](#) &other)  
*Assignment is allowed.*
- [Document](#) ()  
*Make a new empty Document.*
- [~Document](#) ()  
*Destructor.*
- std::string [get\\_value](#) ([Xapian::valueno](#) valueno) const  
*Get value by number.*
- void [add\\_value](#) ([Xapian::valueno](#) valueno, const std::string &value)  
*Add a new value.*
- void [remove\\_value](#) ([Xapian::valueno](#) valueno)  
*Remove any value with the given number.*
- void [clear\\_values](#) ()  
*Remove all values associated with the document.*
- std::string [get\\_data](#) () const  
*Get data stored in the document.*
- void [set\\_data](#) (const std::string &data)  
*Set data stored in the document.*
- void [add\\_posting](#) (const std::string &tname, [Xapian::termpos](#) tpos, [Xapian::termcount](#) wdfinc=1)  
*Add an occurrence of a term at a particular position.*
- void [add\\_term](#) (const std::string &tname, [Xapian::termcount](#) wdfinc=1)  
*Add a term to the document, without positional information.*
- void [add\\_boolean\\_term](#) (const std::string &term)  
*Add a boolean filter term to the document.*

- void `remove_posting` (const std::string &tname, `Xapian::termpos` tpos, `Xapian::termcount` wdfdec=1)  
*Remove a posting of a term from the document.*
- void `remove_term` (const std::string &tname)  
*Remove a term and all postings associated with it.*
- void `clear_terms` ()  
*Remove all terms (and postings) from the document.*
- `Xapian::termcount` `termlist_count` () const  
*The length of the termlist - i.e.*
- `TermIterator` `termlist_begin` () const  
*Iterator for the terms in this document.*
- `TermIterator` `termlist_end` () const  
*Equivalent end iterator for `termlist_begin()`.*
- `Xapian::termcount` `values_count` () const  
*Count the values in this document.*
- `ValueIterator` `values_begin` () const  
*Iterator for the values in this document.*
- `ValueIterator` `values_end` () const  
*Equivalent end iterator for `values_begin()`.*
- `docid` `get_docid` () const  
*Get the document id which is associated with this document (if any).*
- std::string `get_description` () const  
*Return a string describing this object.*

### 7.14.1 Detailed Description

A handle representing a document in a `Xapian` database.

The `Document` class fetches information from the database lazily. Usually this behaviour isn't visible to users (except for the speed benefits), but if the document in the database is modified or deleted, then preexisting `Document` objects may return the old or new versions of data (or throw `Xapian::DocNotFoundError` in the case of deletion).

Since `Database` objects work on a snapshot of the database's state, the situation above can only happen with a `WritableDatabase` object, or if you call `Database::reopen()` on a `Database` object.

We recommend you avoid designs where this behaviour is an issue.



## 7.14.2 Constructor & Destructor Documentation

### 7.14.2.1 Xapian::Document::Document (const Document & *other*)

Copying is allowed.

The internals are reference counted, so copying is cheap.

### 7.14.2.2 Xapian::Document::Document ()

Make a new empty [Document](#).

### 7.14.2.3 Xapian::Document::~~Document ()

Destructor.

## 7.14.3 Member Function Documentation

### 7.14.3.1 void Xapian::Document::add\_boolean\_term (const std::string & *term*) [inline]

Add a boolean filter term to the document.

This method adds *term* to the document with wdf of 0 - this is generally what you want for a term used for boolean filtering as the wdf of such terms is ignored, and it doesn't make sense for them to contribute to the document's length.

If the specified term already indexes this document, this method has no effect.

It is exactly the same as `add_term(term, 0)`.

This method was added in [Xapian 1.0.18](#).

#### Parameters:

*term* The term to add.

### 7.14.3.2 void Xapian::Document::add\_posting (const std::string & *tname*, Xapian::termpos *tpos*, Xapian::termcount *wdfinc* = 1)

Add an occurrence of a term at a particular position.

Multiple occurrences of the term at the same position are represented only once in the positional information, but do increase the wdf.

If the term is not already in the document, it will be added to it.

#### Parameters:

*tname* The name of the term.

*tpos* The position of the term.

*wdfinc* The increment that will be applied to the wdf for this term.

#### 7.14.3.3 void Xapian::Document::add\_term (const std::string & *tname*, Xapian::termcount *wdfinc* = 1)

Add a term to the document, without positional information.

Any existing positional information for the term will be left unmodified.

##### Parameters:

*tname* The name of the term.

*wdfinc* The increment that will be applied to the wdf for this term (default: 1).

#### 7.14.3.4 void Xapian::Document::add\_value (Xapian::valueno *valueno*, const std::string & *value*)

Add a new value.

The new value will replace any existing value with the same number (or if the new value is empty, it will remove any existing value with the same number).

#### 7.14.3.5 void Xapian::Document::clear\_terms ()

Remove all terms (and postings) from the document.

#### 7.14.3.6 void Xapian::Document::clear\_values ()

Remove all values associated with the document.

#### 7.14.3.7 std::string Xapian::Document::get\_data () const

Get data stored in the document.

This is a potentially expensive operation, and shouldn't normally be used in a match decider functor. Put data for use by match deciders in a value instead.

#### 7.14.3.8 std::string Xapian::Document::get\_description () const

Return a string describing this object.

### 7.14.3.9 docid Xapian::Document::get\_docid () const

Get the document id which is associated with this document (if any).

NB If multiple databases are being searched together, then this will be the document id in the individual database, not the merged database!

#### Returns:

If this document came from a database, return the document id in that database. Otherwise, return 0 (in [Xapian 1.0.22/1.2.4](#) or later; prior to this the returned value was uninitialised).

### 7.14.3.10 std::string Xapian::Document::get\_value (Xapian::valueno *valueno*) const

Get value by number.

Returns an empty string if no value with the given number is present in the document.

#### Parameters:

*valueno* The number of the value.

### 7.14.3.11 void Xapian::Document::operator= (const Document & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

### 7.14.3.12 void Xapian::Document::remove\_posting (const std::string & *tname*, Xapian::termpos *tpos*, Xapian::termcount *wdfdec* = 1)

Remove a posting of a term from the document.

Note that the term will still index the document even if all occurrences are removed. To remove a term from a document completely, use [remove\\_term\(\)](#).

#### Parameters:

*tname* The name of the term.

*tpos* The position of the term.

*wdfdec* The decrement that will be applied to the wdf when removing this posting. The wdf will not go below the value of 0.

#### Exceptions:

[Xapian::InvalidArgumentError](#) will be thrown if the term is not at the position specified in the position list for this term in this document.

[Xapian::InvalidArgumentError](#) will be thrown if the term is not in the document

**7.14.3.13 void Xapian::Document::remove\_term (const std::string & *tname*)**

Remove a term and all postings associated with it.

**Parameters:**

*tname* The name of the term.

**Exceptions:**

[\*Xapian::InvalidArgumentError\*](#) will be thrown if the term is not in the document

**7.14.3.14 void Xapian::Document::remove\_value (Xapian::valueno *valueno*)**

Remove any value with the given number.

**7.14.3.15 void Xapian::Document::set\_data (const std::string & *data*)**

Set data stored in the document.

**7.14.3.16 TermIterator Xapian::Document::termlist\_begin () const**

Iterator for the terms in this document.

**7.14.3.17 Xapian::termcount Xapian::Document::termlist\_count () const**

The length of the termlist - i.e.  
the number of different terms which index this document.

**7.14.3.18 TermIterator Xapian::Document::termlist\_end () const** `[inline]`

Equivalent end iterator for [`termlist\_begin\(\)`](#).

**7.14.3.19 ValueIterator Xapian::Document::values\_begin () const**

Iterator for the values in this document.

**7.14.3.20 Xapian::termcount Xapian::Document::values\_count () const**

Count the values in this document.

#### 7.14.3.21 ValueIterator Xapian::Document::values\_end () const

Equivalent end iterator for [values\\_begin\(\)](#).

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

- [xapian/document.h](#)

## 7.15 Xapian::Enquire Class Reference

This class provides an interface to the information retrieval system for the purpose of searching.

### Public Types

- enum **docid\_order** { **ASCENDING** = 1, **DESCENDING** = 0, **DONT\_CARE** = 2 }

### Public Member Functions

- [Enquire](#) (const [Enquire](#) &other)  
*Copying is allowed (and is cheap).*
- void [operator=](#) (const [Enquire](#) &other)  
*Assignment is allowed (and is cheap).*
- [Enquire](#) (const [Database](#) &database, [ErrorHandler](#) \*errorhandler\_=0)  
*Create a [Xapian::Enquire](#) object.*
- [~Enquire](#) ()  
*Close the [Xapian::Enquire](#) object.*
- void [set\\_query](#) (const [Xapian::Query](#) &query, [Xapian::termcount](#) qlen=0)  
*Set the query to run.*
- const [Xapian::Query](#) & [get\\_query](#) () const  
*Get the query which has been set.*
- void [set\\_weighting\\_scheme](#) (const [Weight](#) &weight\_)  
*Set the weighting scheme to use for queries.*
- void [set\\_collapse\\_key](#) ([Xapian::valueno](#) collapse\_key)  
*Set the collapse key to use for queries.*
- void [set\\_docid\\_order](#) (docid\_order order)  
*Set the direction in which documents are ordered by document id in the returned [MSet](#).*
- void [set\\_cutoff](#) ([Xapian::percent](#) percent\_cutoff, [Xapian::weight](#) weight\_cutoff=0)  
*Set the percentage and/or weight cutoffs.*
- void [set\\_sort\\_by\\_relevance](#) ()  
*Set the sorting to be by relevance only.*

- void `set_sort_by_value` (`Xapian::valueno` sort\_key, bool reverse=true)  
*Set the sorting to be by value only.*
- void `set_sort_by_key` (`Xapian::Sorter` \*sorter, bool reverse=true)  
*Set the sorting to be by key generated from values only.*
- void `set_sort_by_value_then_relevance` (`Xapian::valueno` sort\_key, bool reverse=true)  
*Set the sorting to be by value, then by relevance for documents with the same value.*
- void `set_sort_by_key_then_relevance` (`Xapian::Sorter` \*sorter, bool reverse=true)  
*Set the sorting to be by keys generated from values, then by relevance for documents with identical keys.*
- void `set_sort_by_relevance_then_value` (`Xapian::valueno` sort\_key, bool reverse=true)  
*Set the sorting to be by relevance then value.*
- void `set_sort_by_relevance_then_key` (`Xapian::Sorter` \*sorter, bool reverse=true)  
*Set the sorting to be by relevance, then by keys generated from values.*
- `MSet` `get_mset` (`Xapian::doccount` first, `Xapian::doccount` maxitems, `Xapian::doccount` checkatleast=0, const `RSet` \*omrset=0, const `MatchDecider` \*mdecider=0) const  
*Get (a portion of) the match set for the current query.*
- `MSet` `get_mset` (`Xapian::doccount` first, `Xapian::doccount` maxitems, `Xapian::doccount` checkatleast, const `RSet` \*omrset, const `MatchDecider` \*mdecider, const `MatchDecider` \*matchspy) const
- `MSet` `get_mset` (`Xapian::doccount` first, `Xapian::doccount` maxitems, const `RSet` \*omrset, const `MatchDecider` \*mdecider=0) const
- `ESet` `get_eset` (`Xapian::termcount` maxitems, const `RSet` &omrset, int flags=0, double k=1.0, const `Xapian::ExpandDecider` \*edecider=0) const  
*Get the expand set for the given rset.*
- `ESet` `get_eset` (`Xapian::termcount` maxitems, const `RSet` &omrset, const `Xapian::ExpandDecider` \*edecider) const  
*Get the expand set for the given rset.*
- `TermIterator` `get_matching_terms_begin` (`Xapian::docid` did) const  
*Get terms which match a given document, by document id.*
- `TermIterator` `get_matching_terms_end` (`Xapian::docid`) const  
*End iterator corresponding to `get_matching_terms_begin()`.*

- [TermIterator](#) [get\\_matching\\_terms\\_begin](#) (const [MSetIterator](#) &it) const  
*Get terms which match a given document, by match set item.*
- [TermIterator](#) [get\\_matching\\_terms\\_end](#) (const [MSetIterator](#) &) const  
*End iterator corresponding to [get\\_matching\\_terms\\_begin](#)().*
- void [register\\_match\\_decider](#) (const std::string &name, const [MatchDecider](#) \*mdecider=NULL)  
*Register a [MatchDecider](#).*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

## Public Attributes

- [Xapian::Internal::RefCntPtr< Internal >](#) **internal**

## Static Public Attributes

- static const int **INCLUDE\_QUERY\_TERMS** = 1
- static const int **USE\_EXACT\_TERM\_FREQ** = 2
- static const int [include\\_query\\_terms](#) = 1  
*Deprecated in [Xapian](#) 1.0.0, use [INCLUDE\\_QUERY\\_TERMS](#) instead.*
- static const int [use\\_exact\\_termfreq](#) = 2  
*Deprecated in [Xapian](#) 1.0.0, use [USE\\_EXACT\\_TERM\\_FREQ](#) instead.*

### 7.15.1 Detailed Description

This class provides an interface to the information retrieval system for the purpose of searching.

Databases are usually opened lazily, so exceptions may not be thrown where you would expect them to be. You should catch [Xapian::Error](#) exceptions when calling any method in [Xapian::Enquire](#).

#### Exceptions:

[Xapian::InvalidArgumentError](#) will be thrown if an invalid argument is supplied, for example, an unknown database type.



## 7.15.2 Constructor & Destructor Documentation

### 7.15.2.1 Xapian::Enquire::Enquire (const Enquire & *other*)

Copying is allowed (and is cheap).

### 7.15.2.2 Xapian::Enquire::Enquire (const Database & *database*, ErrorHandler \* *errorhandler\_* = 0) [explicit]

Create a [Xapian::Enquire](#) object.

This specification cannot be changed once the [Xapian::Enquire](#) is opened: you must create a new [Xapian::Enquire](#) object to access a different database, or set of databases.

The database supplied must have been initialised (ie, must not be the result of calling the [Database::Database\(\)](#) constructor). If you need to handle a situation where you have no index gracefully, a database created with [InMemory::open\(\)](#) can be passed here, which represents a completely empty database.

#### Parameters:

*database* Specification of the database or databases to use.

*errorhandler\_* A pointer to the error handler to use. Ownership of the object pointed to is not assumed by the [Xapian::Enquire](#) object - the user should delete the [Xapian::ErrorHandler](#) object after the [Xapian::Enquire](#) object is deleted. To use no error handler, this parameter should be 0.

#### Exceptions:

[Xapian::InvalidArgumentError](#) will be thrown if an initialised [Database](#) object is supplied.

### 7.15.2.3 Xapian::Enquire::~~Enquire ()

Close the [Xapian::Enquire](#) object.

## 7.15.3 Member Function Documentation

### 7.15.3.1 std::string Xapian::Enquire::get\_description () const

Return a string describing this object.

### 7.15.3.2 ESet Xapian::Enquire::get\_eset (Xapian::termcount *maxitems*, const RSet & *omrset*, const Xapian::ExpandDecider \* *edecider*) const [inline]

Get the expand set for the given rset.

**Parameters:**

*maxitems* the maximum number of items to return.  
*omrset* the relevance set to use when performing the expand operation.  
*edecider* a decision functor to use to decide whether a given term should be put in the [ESet](#)

**Returns:**

An [ESet](#) object containing the results of the expand.

**Exceptions:**

[\*Xapian::InvalidArgumentError\*](#) See class documentation.

#### 7.15.3.3 **ESet Xapian::Enquire::get\_eset (Xapian::termcount *maxitems*, const RSet & *omrset*, int *flags* = 0, double *k* = 1.0, const Xapian::ExpandDecider \* *edecider* = 0) const**

Get the expand set for the given rset.

**Parameters:**

*maxitems* the maximum number of items to return.  
*omrset* the relevance set to use when performing the expand operation.  
*flags* zero or more of these values |-ed together:

- Xapian::Enquire::INCLUDE\_QUERY\_TERMS query terms may be returned from expand
- Xapian::Enquire::USE\_EXACT\_TERM\_FREQ for multi dbs, calculate the exact termfreq; otherwise an approximation is used which can greatly improve efficiency, but still returns good results.

*k* the parameter k in the query expansion algorithm (default is 1.0)  
*edecider* a decision functor to use to decide whether a given term should be put in the [ESet](#)

**Returns:**

An [ESet](#) object containing the results of the expand.

**Exceptions:**

[\*Xapian::InvalidArgumentError\*](#) See class documentation.

#### 7.15.3.4 **TermIterator Xapian::Enquire::get\_matching\_terms\_begin (const MSetIterator & *it*) const**

Get terms which match a given document, by match set item.

This method returns the terms in the current query which match the given document.

If the underlying database has suitable support, using this call (rather than passing a [Xapian::docid](#)) will enable the system to ensure that the correct data is returned, and that the document has not been deleted or changed since the query was performed.

**Parameters:**

*it* The iterator for which to retrieve the matching terms.

**Returns:**

An iterator returning the terms which match the document. The terms will be returned (as far as this makes any sense) in the same order as the terms in the query. Terms will not occur more than once, even if they do in the query.

**Exceptions:**

[Xapian::InvalidArgumentError](#) See class documentation.

[Xapian::DocNotFoundError](#) The document specified could not be found in the database.

### 7.15.3.5 TermIterator Xapian::Enquire::get\_matching\_terms\_begin (Xapian::docid *did*) const

Get terms which match a given document, by document id.

This method returns the terms in the current query which match the given document.

It is possible for the document to have been removed from the database between the time it is returned in an [MSet](#), and the time that this call is made. If possible, you should specify an [MSetIterator](#) instead of a [Xapian::docid](#), since this will enable database backends with suitable support to prevent this occurring.

Note that a query does not need to have been run in order to make this call.

**Parameters:**

*did* The document id for which to retrieve the matching terms.

**Returns:**

An iterator returning the terms which match the document. The terms will be returned (as far as this makes any sense) in the same order as the terms in the query. Terms will not occur more than once, even if they do in the query.

**Exceptions:**

[Xapian::InvalidArgumentError](#) See class documentation.

[Xapian::DocNotFoundError](#) The document specified could not be found in the database.

#### 7.15.3.6 `TermIterator Xapian::Enquire::get_matching_terms_end (const MSetIterator &) const` [inline]

End iterator corresponding to [get\\_matching\\_terms\\_begin\(\)](#).

#### 7.15.3.7 `TermIterator Xapian::Enquire::get_matching_terms_end (Xapian::docid) const` [inline]

End iterator corresponding to [get\\_matching\\_terms\\_begin\(\)](#).

#### 7.15.3.8 `MSet Xapian::Enquire::get_mset (Xapian::doccount first, Xapian::doccount maxitems, Xapian::doccount checkatleast = 0, const RSet * omrset = 0, const MatchDecider * mdecider = 0) const`

Get (a portion of) the match set for the current query.

##### Parameters:

***first*** the first item in the result set to return. A value of zero corresponds to the first item returned being that with the highest score. A value of 10 corresponds to the first 10 items being ignored, and the returned items starting at the eleventh.

***maxitems*** the maximum number of items to return. If you want all matches, then you can pass the result of calling [get\\_doccount\(\)](#) on the [Database](#) object (though if you are doing this so you can filter results, you are likely to get much better performance by using Xapian's match-time filtering features instead). You can pass 0 for *maxitems* which will give you an empty [MSet](#) with valid statistics (such as [get\\_matches\\_estimated\(\)](#)) calculated without looking at any postings, which is very quick, but means the estimates may be more approximate and the bounds may be much looser.

***checkatleast*** the minimum number of items to check. Because the matcher optimises, it won't consider every document which might match, so the total number of matches is estimated. Setting *checkatleast* forces it to consider at least this many matches and so allows for reliable paging links.

***omrset*** the relevance set to use when performing the query.

***mdecider*** a decision functor to use to decide whether a given document should be put in the [MSet](#).

***matchspy*** a decision functor to use to decide whether a given document should be put in the [MSet](#). The *matchspy* is applied to every document which is a potential candidate for the [MSet](#), so if there are *checkatleast* or more such documents, the *matchspy* will see at least *checkatleast*. The *mdecider* is assumed to be a relatively expensive test so may be applied in a lazier fashion.

##### Returns:

A [Xapian::MSet](#) object containing the results of the query.

**Exceptions:**

[\*Xapian::InvalidArgumentError\*](#) See class documentation.

**7.15.3.9 const Xapian::Query& Xapian::Enquire::get\_query () const**

Get the query which has been set.

This is only valid after [set\\_query\(\)](#) has been called.

**Exceptions:**

[\*Xapian::InvalidArgumentError\*](#) will be thrown if query has not yet been set.

**7.15.3.10 void Xapian::Enquire::operator= (const Enquire & other)**

Assignment is allowed (and is cheap).

**7.15.3.11 void Xapian::Enquire::register\_match\_decider (const std::string & name, const MatchDecider \* mdecider = NULL)**

Register a [MatchDecider](#).

This is used to associate a name with a matchdecider.

**Deprecated**

This method is deprecated. It was added long ago with the intention that it would allow the remote backend to support use of [MatchDecider](#) objects, but there's a better approach.

**Parameters:**

*name* The name to register this matchdecider as.

*mdecider* The matchdecider. If omitted, then remove any matchdecider registered with this name.

**7.15.3.12 void Xapian::Enquire::set\_collapse\_key (Xapian::valueno collapse\_key)**

Set the collapse key to use for queries.

**Parameters:**

*collapse\_key* value number to collapse on - at most one [MSet](#) entry with each particular value will be returned.

The entry returned will be the best entry with that particular value (highest weight or highest sorting key).

An example use might be to create a value for each document containing an MD5 hash of the document contents. Then duplicate documents from different sources can be eliminated at search time (it's better to eliminate duplicates at index time, but this may not be always be possible - for example the search may be over more than one [Xapian](#) database).

Another use is to group matches in a particular category (e.g. you might collapse a mailing list search on the Subject: so that there's only one result per discussion thread). In this case you can use `get_collapse_count()` to give the user some idea how many other results there are. And if you index the Subject: as a boolean term as well as putting it in a value, you can offer a link to a non-collapsed search restricted to that thread using a boolean filter.

(default is [Xapian::BAD\\_VALUENO](#) which means no collapsing).

#### 7.15.3.13 `void Xapian::Enquire::set_cutoff (Xapian::percent percent_cutoff, Xapian::weight weight_cutoff = 0)`

Set the percentage and/or weight cutoffs.

##### Parameters:

*percent\_cutoff* Minimum percentage score for returned documents. If a document has a lower percentage score than this, it will not appear in the [MSet](#). If your intention is to return only matches which contain all the terms in the query, then it's more efficient to use [Xapian::Query::OP\\_AND](#) instead of [Xapian::Query::OP\\_OR](#) in the query than to use `set_cutoff(100)`. (default 0 => no percentage cut-off).

*weight\_cutoff* Minimum weight for a document to be returned. If a document has a lower score than this, it will not appear in the [MSet](#). It is usually only possible to choose an appropriate weight for cutoff based on the results of a previous run of the same query; this is thus mainly useful for alerting operations. The other potential use is with a user specified weighting scheme. (default 0 => no weight cut-off).

#### 7.15.3.14 `void Xapian::Enquire::set_docid_order (docid_order order)`

Set the direction in which documents are ordered by document id in the returned [MSet](#).

This order only has an effect on documents which would otherwise have equal rank. For a weighted probabilistic match with no sort value, this means documents with equal weight. For a boolean match, with no sort value, this means all documents. And if a sort value is used, this means documents with equal sort value (and also equal weight if ordering on relevance after the sort).

##### Parameters:

*order* This can be:

- Xapian::Enquire::ASCENDING docids sort in ascending order (default)
- Xapian::Enquire::DESCENDING docids sort in descending order
- Xapian::Enquire::DONT\_CARE docids sort in whatever order is most efficient for the backend

Note: If you add documents in strict date order, then a boolean search - i.e. `set_weighting_scheme(Xapian::BoolWeight())` - with `set_docid_order(Xapian::Enquire::DESCENDING)` is a very efficient way to perform "sort by date, newest first".

#### 7.15.3.15 `void Xapian::Enquire::set_query (const Xapian::Query & query, Xapian::termcount qlen = 0)`

Set the query to run.

##### Parameters:

**query** the new query to run.

**qlen** the query length to use in weight calculations - by default the sum of the wqf of all terms is used.

#### 7.15.3.16 `void Xapian::Enquire::set_sort_by_key (Xapian::Sorter * sorter, bool reverse = true)`

Set the sorting to be by key generated from values only.

##### Parameters:

**sorter** The functor to use for generating keys.

**reverse** If true, reverses the sort order. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

#### 7.15.3.17 `void Xapian::Enquire::set_sort_by_key_then_relevance (Xapian::Sorter * sorter, bool reverse = true)`

Set the sorting to be by keys generated from values, then by relevance for documents with identical keys.

##### Parameters:

**sorter** The functor to use for generating keys.

**reverse** If true, reverses the sort order. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

### 7.15.3.18 void Xapian::Enquire::set\_sort\_by\_relevance ()

Set the sorting to be by relevance only.

This is the default.

### 7.15.3.19 void Xapian::Enquire::set\_sort\_by\_relevance\_then\_key (Xapian::Sorter \* *sorter*, bool *reverse* = true)

Set the sorting to be by relevance, then by keys generated from values.

Note that with the default BM25 weighting scheme parameters, non-identical documents will rarely have the same weight, so this setting will give very similar results to [set\\_sort\\_by\\_relevance\(\)](#). It becomes more useful with particular BM25 parameter settings (e.g. BM25Weight(1,0,1,0,0)) or custom weighting schemes.

#### Parameters:

*sorter* The functor to use for generating keys.

*reverse* If true, reverses the sort order. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

### 7.15.3.20 void Xapian::Enquire::set\_sort\_by\_relevance\_then\_value (Xapian::valueno *sort\_key*, bool *reverse* = true)

Set the sorting to be by relevance then value.

Note that sorting by values uses a string comparison, so to use this to sort by a numeric value you'll need to store the numeric values in a manner which sorts appropriately. For example, you could use [Xapian::sortable\\_serialise\(\)](#) (which works for floating point numbers as well as integers), or store numbers padded with leading zeros or spaces, or with the number of digits prepended.

Note that with the default BM25 weighting scheme parameters, non-identical documents will rarely have the same weight, so this setting will give very similar results to [set\\_sort\\_by\\_relevance\(\)](#). It becomes more useful with particular BM25 parameter settings (e.g. BM25Weight(1,0,1,0,0)) or custom weighting schemes.

#### Parameters:

*sort\_key* value number to sort on.

*reverse* If true, reverses the sort order. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

### 7.15.3.21 void Xapian::Enquire::set\_sort\_by\_value (Xapian::valueno *sort\_key*, bool *reverse* = true)

Set the sorting to be by value only.



Note that sorting by values uses a string comparison, so to use this to sort by a numeric value you'll need to store the numeric values in a manner which sorts appropriately. For example, you could use [Xapian::sortable\\_serialise\(\)](#) (which works for floating point numbers as well as integers), or store numbers padded with leading zeros or spaces, or with the number of digits prepended.

**Parameters:**

*sort\_key* value number to sort on.

*reverse* If true, reverses the sort order. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

**7.15.3.22 void Xapian::Enquire::set\_sort\_by\_value\_then\_relevance  
(Xapian::value no\_sort\_key, bool reverse = true)**

Set the sorting to be by value, then by relevance for documents with the same value.

Note that sorting by values uses a string comparison, so to use this to sort by a numeric value you'll need to store the numeric values in a manner which sorts appropriately. For example, you could use [Xapian::sortable\\_serialise\(\)](#) (which works for floating point numbers as well as integers), or store numbers padded with leading zeros or spaces, or with the number of digits prepended.

**Parameters:**

*sort\_key* value number to sort on.

*reverse* If true, reverses the sort order. (default true, but this default is confusing and deprecated in 1.1.0, so we recommend specifying this parameter explicitly).

**7.15.3.23 void Xapian::Enquire::set\_weighting\_scheme (const Weight &weight\_)**

Set the weighting scheme to use for queries.

**Parameters:**

*weight\_* the new weighting scheme. If no weighting scheme is specified, the default is BM25 with the default parameters.

**7.15.4 Member Data Documentation****7.15.4.1 const int Xapian::Enquire::include\_query\_terms = 1 [static]**

Deprecated in [Xapian](#) 1.0.0, use INCLUDE\_QUERY\_TERMS instead.

**7.15.4.2** `const int Xapian::Enquire::use_exact_termfreq = 2` `[static]`

Deprecated in [Xapian 1.0.0](#), use `USE_EXACT_TERMFREQ` instead.

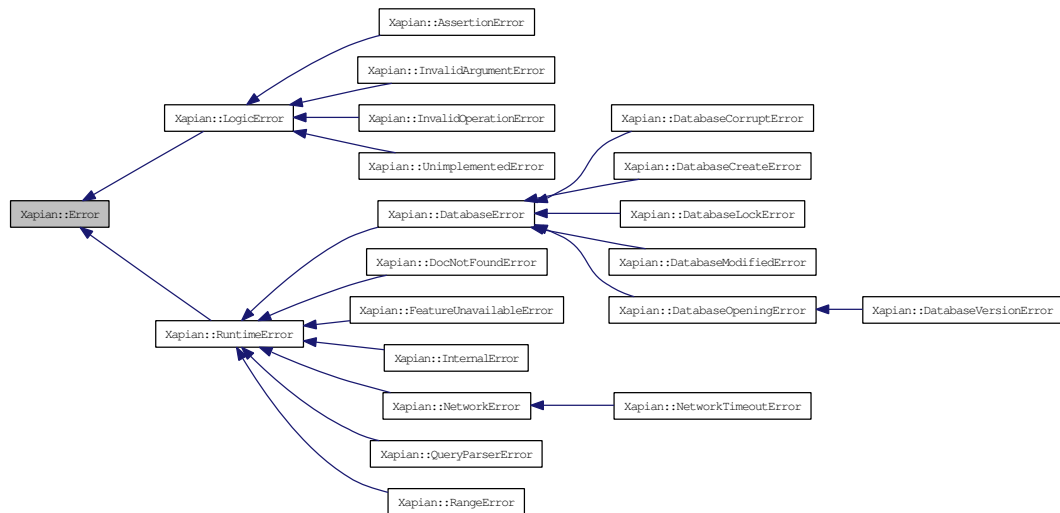
The documentation for this class was generated from the following file:

- [xapian/enquire.h](#)

## 7.16 Xapian::Error Class Reference

All exceptions thrown by [Xapian](#) are subclasses of [Xapian::Error](#).

Inheritance diagram for [Xapian::Error](#):



### Public Member Functions

- `const char * get\_type () const`  
*The type of this error (e.g. "DocNotFoundError").*
- `const std::string & get\_msg () const`  
*Message giving details of the error, intended for human consumption.*
- `const std::string & get\_context () const`  
*Optional context information.*
- `const char * get\_error\_string () const`  
*Returns any system error string associated with this exception.*
- `int get\_errno () const`  
*Optional value of 'errno' associated with this error.*
- `std::string get\_description () const`  
*Return a string describing this object.*

### 7.16.1 Detailed Description

All exceptions thrown by [Xapian](#) are subclasses of [Xapian::Error](#).

This class can not be instantiated directly - instead a subclass should be used.

### 7.16.2 Member Function Documentation

#### 7.16.2.1 `const std::string& Xapian::Error::get_context () const` [inline]

Optional context information.

This context is intended for use by [Xapian::ErrorHandler](#) (for example so it can know which remote server is unreliable and report the problem and remove that server from those being searched). But it's typically a plain-text string, and so also fit for human consumption.

#### 7.16.2.2 `std::string Xapian::Error::get_description () const`

Return a string describing this object.

#### 7.16.2.3 `int Xapian::Error::get_errno () const` [inline]

Optional value of 'errno' associated with this error.

If no 'errno' value is associated, returns 0. If the returned value is negative, it's a platform-specific error code (on UNIX, `-h_errno`; on MS Windows, `-GetLastError()`).

#### Deprecated

This method is deprecated, because errno values aren't portable between platforms, so we can't serialise them when passing exceptions from a remote server to a client. Use the [get\\_error\\_string\(\)](#) method instead.

#### 7.16.2.4 `const char* Xapian::Error::get_error_string () const`

Returns any system error string associated with this exception.

The system error string may come from `errno`, `h_errno` (on UNIX), or `GetLastError()` (on MS Windows). If there is no associated system error string, `NULL` is returned.

#### 7.16.2.5 `const std::string& Xapian::Error::get_msg () const` [inline]

Message giving details of the error, intended for human consumption.

**7.16.2.6** `const char* Xapian::Error::get_type () const` `[inline]`

The type of this error (e.g. "DocNotFoundError").

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

- [xapian/error.h](#)

## 7.17 Xapian::ErrorHandler Class Reference

Decide if a [Xapian::Error](#) exception should be ignored.

### Public Member Functions

- [ErrorHandler](#) ()  
*Default constructor.*
- virtual [~ErrorHandler](#) ()  
*We require a virtual destructor because we have virtual methods.*
- void [operator\(\)](#) ([Xapian::Error](#) &error)  
*Handle a [Xapian::Error](#) object.*

#### 7.17.1 Detailed Description

Decide if a [Xapian::Error](#) exception should be ignored.

You can create your own subclass of this class and pass in an instance of it when you construct a [Xapian::Enquire](#) object. [Xapian::Error](#) exceptions which happen during the match process are passed to this object and it can decide whether they should propagate or whether [Enquire](#) should attempt to continue.

The motivation is to allow searching over remote databases to handle a remote server which has died (both to allow results to be returned, and also so that such errors can be logged and dead servers temporarily removed from use).

#### 7.17.2 Constructor & Destructor Documentation

##### 7.17.2.1 Xapian::ErrorHandler::ErrorHandler () [inline]

Default constructor.

##### 7.17.2.2 virtual Xapian::ErrorHandler::~~ErrorHandler () [virtual]

We require a virtual destructor because we have virtual methods.

#### 7.17.3 Member Function Documentation

##### 7.17.3.1 void Xapian::ErrorHandler::operator() (Xapian::Error & error)

Handle a [Xapian::Error](#) object.

This method is called when a [Xapian::Error](#) object is thrown and caught inside [Enquire](#). If this is the first [ErrorHandler](#) that the [Error](#) has been passed to, then the `handle_error()` virtual method is called, which allows the API user to decide how to handle the error.

**Parameters:**

*error* The [Xapian::Error](#) object under consideration.

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

- [xapian/errorhandler.h](#)

## 7.18 Xapian::ESet Class Reference

Class representing an ordered set of expand terms (an [ESet](#)).

### Public Member Functions

- [ESet](#) ()  
*Construct an empty [ESet](#).*
- [~ESet](#) ()  
*Destructor.*
- [ESet](#) (const [ESet](#) &other)  
*Copying is allowed (and is cheap).*
- void [operator=](#) (const [ESet](#) &other)  
*Assignment is allowed (and is cheap).*
- [Xapian::termcount](#) [get\\_ebound](#) () const  
*A lower bound on the number of terms which are in the full set of results of the expand.*
- [Xapian::termcount](#) [size](#) () const  
*The number of terms in this E-Set.*
- [Xapian::termcount](#) [max\\_size](#) () const  
*Required to allow use as an STL container.*
- bool [empty](#) () const  
*Test if this E-Set is empty.*
- void [swap](#) ([ESet](#) &other)  
*Swap the E-Set we point to with another.*
- [ESetIterator](#) [begin](#) () const  
*Iterator for the terms in this E-Set.*
- [ESetIterator](#) [end](#) () const  
*End iterator corresponding to [begin\(\)](#).*
- [ESetIterator](#) [back](#) () const  
*Iterator pointing to the last element of this E-Set.*
- [ESetIterator](#) [operator\[\]](#) ([Xapian::termcount](#) i) const  
*This returns the term at position i in this E-Set.*
- std::string [get\\_description](#) () const



*Return a string describing this object.*

## Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

### 7.18.1 Detailed Description

Class representing an ordered set of expand terms (an [ESet](#)).

This set represents the results of an expand operation, which is performed by [Xapian::Enquire::get\\_eset\(\)](#).

### 7.18.2 Constructor & Destructor Documentation

#### 7.18.2.1 Xapian::ESet::ESet ()

Construct an empty [ESet](#).

#### 7.18.2.2 Xapian::ESet::~~ESet ()

Destructor.

#### 7.18.2.3 Xapian::ESet::ESet (const ESet & *other*)

Copying is allowed (and is cheap).

### 7.18.3 Member Function Documentation

#### 7.18.3.1 ESetIterator Xapian::ESet::back () const

Iterator pointing to the last element of this E-Set.

#### 7.18.3.2 ESetIterator Xapian::ESet::begin () const

Iterator for the terms in this E-Set.

#### 7.18.3.3 bool Xapian::ESet::empty () const

Test if this E-Set is empty.

**7.18.3.4 ESetIterator Xapian::ESet::end () const**

End iterator corresponding to [begin\(\)](#).

**7.18.3.5 std::string Xapian::ESet::get\_description () const**

Return a string describing this object.

**7.18.3.6 Xapian::termcount Xapian::ESet::get\_ebound () const**

A lower bound on the number of terms which are in the full set of results of the expand.  
This will be greater than or equal to [size\(\)](#)

**7.18.3.7 Xapian::termcount Xapian::ESet::max\_size () const** `[inline]`

Required to allow use as an STL container.

**7.18.3.8 void Xapian::ESet::operator= (const ESet & *other*)**

Assignment is allowed (and is cheap).

**7.18.3.9 ESetIterator Xapian::ESet::operator[] (Xapian::termcount *i*) const**

This returns the term at position *i* in this E-Set.

**7.18.3.10 Xapian::termcount Xapian::ESet::size () const**

The number of terms in this E-Set.

**7.18.3.11 void Xapian::ESet::swap (ESet & *other*)**

Swap the E-Set we point to with another.

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

- [xapian/enquire.h](#)

## 7.19 Xapian::ESetIterator Class Reference

Iterate through terms in the [ESet](#).

### Public Types

- typedef std::bidirectional\_iterator\_tag [iterator\\_category](#)  
*Allow use as an STL iterator.*
- typedef std::string [value\\_type](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::termcount\\_diff](#) [difference\\_type](#)  
*Allow use as an STL iterator.*
- typedef std::string \* [pointer](#)  
*Allow use as an STL iterator.*
- typedef std::string & [reference](#)  
*Allow use as an STL iterator.*

### Public Member Functions

- [ESetIterator](#) ()  
*Create an uninitialised iterator; this cannot be used, but is convenient syntactically.*
- [ESetIterator](#) (const [ESetIterator](#) &other)  
*Copying is allowed (and is cheap).*
- void [operator=](#) (const [ESetIterator](#) &other)  
*Assignment is allowed (and is cheap).*
- [ESetIterator](#) & [operator++](#) ()  
*Advance the iterator.*
- [ESetIterator](#) [operator++](#) (int)  
*Advance the iterator (postfix variant).*
- [ESetIterator](#) & [operator--](#) ()  
*Decrement the iterator.*
- [ESetIterator](#) [operator--](#) (int)  
*Decrement the iterator (postfix variant).*
- const std::string & [operator\\*](#) () const

*Get the term for the current position.*

- `Xapian::weight get_weight () const`  
*Get the weight of the term at the current position.*
- `std::string get_description () const`  
*Return a string describing this object.*

## Friends

- `bool operator== (const ESetIterator &a, const ESetIterator &b)`  
*Equality test for `ESetIterator` objects.*
- `bool operator!= (const ESetIterator &a, const ESetIterator &b)`  
*Inequality test for `ESetIterator` objects.*

## 7.19.1 Detailed Description

Iterate through terms in the `ESet`.

## 7.19.2 Member Typedef Documentation

### 7.19.2.1 `typedef Xapian::termcount_diff Xapian::ESetIterator::difference_type`

Allow use as an STL iterator.

### 7.19.2.2 `typedef std::bidirectional_iterator_tag Xapian::ESetIterator::iterator_category`

Allow use as an STL iterator.

### 7.19.2.3 `typedef std::string* Xapian::ESetIterator::pointer`

Allow use as an STL iterator.

### 7.19.2.4 `typedef std::string& Xapian::ESetIterator::reference`

Allow use as an STL iterator.

### 7.19.2.5 `typedef std::string Xapian::ESetIterator::value_type`

Allow use as an STL iterator.

### 7.19.3 Constructor & Destructor Documentation

#### 7.19.3.1 Xapian::ESetIterator::ESetIterator () [inline]

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

#### 7.19.3.2 Xapian::ESetIterator::ESetIterator (const ESetIterator & *other*) [inline]

Copying is allowed (and is cheap).

### 7.19.4 Member Function Documentation

#### 7.19.4.1 std::string Xapian::ESetIterator::get\_description () const

Return a string describing this object.

#### 7.19.4.2 Xapian::weight Xapian::ESetIterator::get\_weight () const

Get the weight of the term at the current position.

#### 7.19.4.3 const std::string& Xapian::ESetIterator::operator\* () const

Get the term for the current position.

#### 7.19.4.4 ESetIterator Xapian::ESetIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

#### 7.19.4.5 ESetIterator& Xapian::ESetIterator::operator++ () [inline]

Advance the iterator.

#### 7.19.4.6 ESetIterator Xapian::ESetIterator::operator-- (int) [inline]

Decrement the iterator (postfix variant).

#### 7.19.4.7 ESetIterator& Xapian::ESetIterator::operator-- () [inline]

Decrement the iterator.

**7.19.4.8** `void Xapian::ESetIterator::operator= (const ESetIterator & other)`  
[inline]

Assignment is allowed (and is cheap).

## 7.19.5 Friends And Related Function Documentation

**7.19.5.1** `bool operator!= (const ESetIterator & a, const ESetIterator & b)`  
[friend]

Inequality test for [ESetIterator](#) objects.

**7.19.5.2** `bool operator== (const ESetIterator & a, const ESetIterator & b)`  
[friend]

Equality test for [ESetIterator](#) objects.

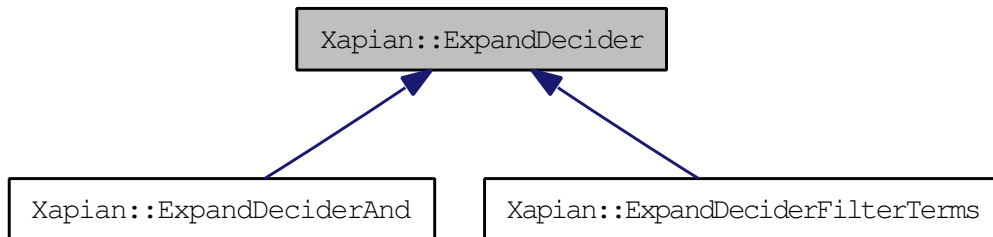
The documentation for this class was generated from the following file:

- [xapian/enquire.h](#)

## 7.20 Xapian::ExpandDecider Class Reference

Virtual base class for expand decider functor.

Inheritance diagram for Xapian::ExpandDecider:



### Public Member Functions

- virtual bool [operator\(\)](#) (const std::string &term) const =0  
*Do we want this term in the [ESet](#)?*
- virtual [~ExpandDecider](#) ()  
*Virtual destructor, because we have virtual methods.*

#### 7.20.1 Detailed Description

Virtual base class for expand decider functor.

#### 7.20.2 Constructor & Destructor Documentation

##### 7.20.2.1 virtual Xapian::ExpandDecider::~~ExpandDecider () [virtual]

Virtual destructor, because we have virtual methods.

#### 7.20.3 Member Function Documentation

##### 7.20.3.1 virtual bool Xapian::ExpandDecider::operator() (const std::string &term) const [pure virtual]

Do we want this term in the [ESet](#)?

Implemented in [Xapian::ExpandDeciderAnd](#), and [Xapian::ExpandDeciderFilterTerms](#).

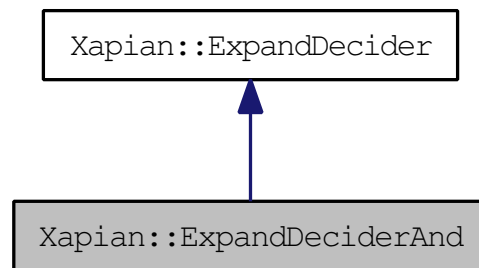
The documentation for this class was generated from the following file:

- [xapian/expanddecider.h](#)

## 7.21 Xapian::ExpandDeciderAnd Class Reference

[ExpandDecider](#) subclass which rejects terms using two ExpandDeciders.

Inheritance diagram for Xapian::ExpandDeciderAnd:



### Public Member Functions

- [ExpandDeciderAnd](#) (const [ExpandDecider](#) &first\_, const [ExpandDecider](#) &second\_)

*Terms will be checked with first, and if accepted, then checked with second.*

- [ExpandDeciderAnd](#) (const [ExpandDecider](#) \*first\_, const [ExpandDecider](#) \*second\_)

*Compatibility method.*

- virtual bool [operator\(\)](#) (const std::string &term) const

*Do we want this term in the [ESet](#)?*

### 7.21.1 Detailed Description

[ExpandDecider](#) subclass which rejects terms using two ExpandDeciders.

Terms are only accepted if they are accepted by both of the specified [ExpandDecider](#) objects.

### 7.21.2 Constructor & Destructor Documentation

- #### 7.21.2.1 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const [ExpandDecider](#) &first\_, const [ExpandDecider](#) &second\_) [inline]

Terms will be checked with *first*, and if accepted, then checked with *second*.



**7.21.2.2** `Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider * first_, const ExpandDecider * second_)` `[inline]`

Compatibility method.

### 7.21.3 Member Function Documentation

**7.21.3.1** `virtual bool Xapian::ExpandDeciderAnd::operator() (const std::string & term) const` `[virtual]`

Do we want this term in the [ESet](#)?

Implements [Xapian::ExpandDecider](#).

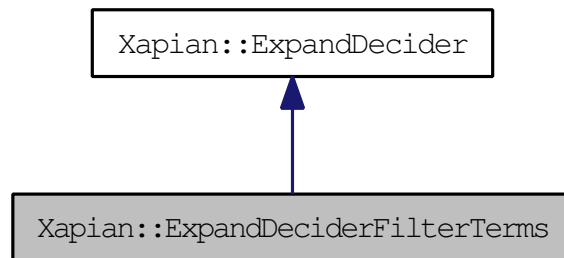
The documentation for this class was generated from the following file:

- [xapian/expanddecider.h](#)

## 7.22 Xapian::ExpandDeciderFilterTerms Class Reference

[ExpandDecider](#) subclass which rejects terms in a specified list.

Inheritance diagram for Xapian::ExpandDeciderFilterTerms:



### Public Member Functions

- template<class Iterator >  
[ExpandDeciderFilterTerms](#) (Iterator reject\_begin, Iterator reject\_end)  
*The two iterators specify a list of terms to be rejected.*
- virtual bool [operator\(\)](#) (const std::string &term) const  
*Do we want this term in the [ESet](#)?*

### 7.22.1 Detailed Description

[ExpandDecider](#) subclass which rejects terms in a specified list.

[ExpandDeciderFilterTerms](#) provides an easy way to filter out terms from a fixed list when generating an [ESet](#).

### 7.22.2 Constructor & Destructor Documentation

- #### 7.22.2.1 template<class Iterator > Xapian::ExpandDeciderFilterTerms::ExpandDeciderFilterTerms (Iterator reject\_begin, Iterator reject\_end) [inline]

The two iterators specify a list of terms to be rejected.

*reject\_begin* and *reject\_end* can be any input\_iterator type which returns std::string or char \* (e.g. [TermIterator](#) or char \*\*).

### 7.22.3 Member Function Documentation

**7.22.3.1** `virtual bool Xapian::ExpandDeciderFilterTerms::operator() (const std::string & term) const` [virtual]

Do we want this term in the [ESet](#)?

Implements [Xapian::ExpandDecider](#).

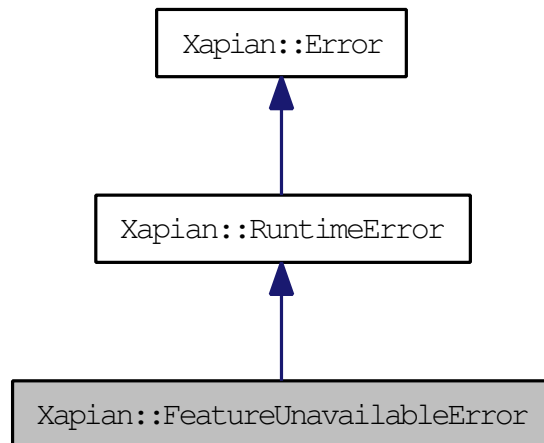
The documentation for this class was generated from the following file:

- [xapian/expanddecider.h](#)

## 7.23 Xapian::FeatureUnavailableError Class Reference

Indicates an attempt to use a feature which is unavailable.

Inheritance diagram for Xapian::FeatureUnavailableError:



### Public Member Functions

- [FeatureUnavailableError](#) (const std::string &msg\_, const std::string &context\_ = "", int errno\_ = 0)  
*General purpose constructor which allows setting errno.*
- [FeatureUnavailableError](#) (const std::string &msg\_, int errno\_)  
*Construct from message and errno value.*

#### 7.23.1 Detailed Description

Indicates an attempt to use a feature which is unavailable.

Typically a feature is unavailable because it wasn't compiled in, or because it requires other software or facilities which aren't available.

#### 7.23.2 Constructor & Destructor Documentation

- ##### 7.23.2.1 Xapian::FeatureUnavailableError::FeatureUnavailableError (const std::string &msg\_, const std::string &context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.23.2.2 Xapian::FeatureUnavailableError::FeatureUnavailableError (const std::string & msg\_, int errno\_) [inline]

Construct from message and errno value.

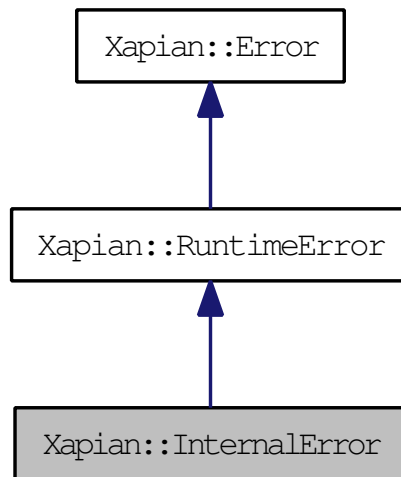
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.24 Xapian::InternalError Class Reference

[InternalError](#) indicates a runtime problem of some sort.

Inheritance diagram for Xapian::InternalError:



### Public Member Functions

- [InternalError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [InternalError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.24.1 Detailed Description

[InternalError](#) indicates a runtime problem of some sort.

### 7.24.2 Constructor & Destructor Documentation

- #### 7.24.2.1 Xapian::InternalError::InternalError (const std::string &msg\_, const std::string &context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.24.2.2 Xapian::InternalError::InternalError (const std::string & *msg\_*, int *errno\_*) [inline]

Construct from message and errno value.

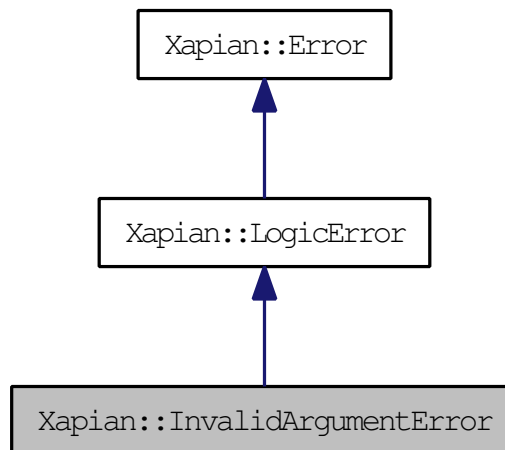
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.25 Xapian::InvalidArgumentError Class Reference

[InvalidArgumentError](#) indicates an invalid parameter value was passed to the API.

Inheritance diagram for Xapian::InvalidArgumentError:



### Public Member Functions

- [InvalidArgumentError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [InvalidArgumentError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.25.1 Detailed Description

[InvalidArgumentError](#) indicates an invalid parameter value was passed to the API.

### 7.25.2 Constructor & Destructor Documentation

- #### 7.25.2.1 Xapian::InvalidArgumentError::InvalidArgumentError (const std::string & msg\_, const std::string & context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.



### 7.25.2.2 Xapian::InvalidArgumentError::InvalidArgumentError (const std::string & msg\_, int errno\_) [inline]

Construct from message and errno value.

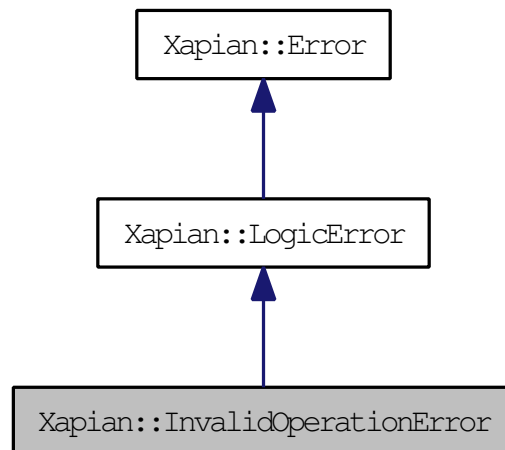
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.26 Xapian::InvalidOperationError Class Reference

[InvalidOperationError](#) indicates the API was used in an invalid way.

Inheritance diagram for Xapian::InvalidOperationError:



### Public Member Functions

- [InvalidOperationError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [InvalidOperationError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.26.1 Detailed Description

[InvalidOperationError](#) indicates the API was used in an invalid way.

### 7.26.2 Constructor & Destructor Documentation

- #### 7.26.2.1 Xapian::InvalidOperationError::InvalidOperationError (const std::string & msg\_, const std::string & context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.26.2.2 Xapian::InvalidOperationError::InvalidOperationError (const std::string & msg\_, int errno\_) [inline]

Construct from message and errno value.

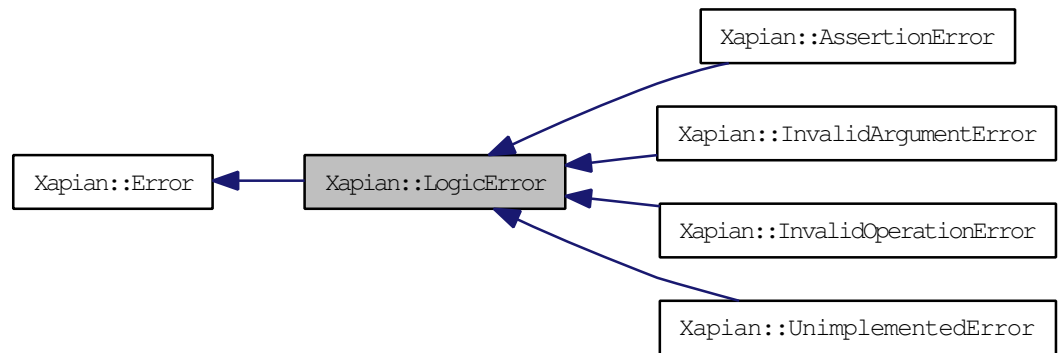
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.27 Xapian::LogicError Class Reference

The base class for exceptions indicating errors in the program logic.

Inheritance diagram for Xapian::LogicError:



### 7.27.1 Detailed Description

The base class for exceptions indicating errors in the program logic.

A subclass of [LogicError](#) will be thrown if [Xapian](#) detects a violation of a class invariant or a logical precondition or postcondition, etc.

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

- [xapian/error.h](#)

## 7.28 Xapian::MatchDecider Class Reference

Base class for matcher decision functor.

### Public Member Functions

- virtual bool [operator\(\)](#) (const [Xapian::Document](#) &doc) const =0  
*Decide whether we want this document to be in the [MSet](#).*
- virtual [~MatchDecider](#) ()  
*Destructor.*

### 7.28.1 Detailed Description

Base class for matcher decision functor.

### 7.28.2 Constructor & Destructor Documentation

#### 7.28.2.1 virtual Xapian::MatchDecider::~~MatchDecider () [virtual]

Destructor.

### 7.28.3 Member Function Documentation

#### 7.28.3.1 virtual bool Xapian::MatchDecider::operator() (const Xapian::Document & doc) const [pure virtual]

Decide whether we want this document to be in the [MSet](#).

Return true if the document is acceptable, or false if the document should be excluded from the [MSet](#).

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

- [xapian/enquire.h](#)

## 7.29 Xapian::MSet Class Reference

A match set ([MSet](#)).

### Public Types

- typedef [MSetIterator](#) [value\\_type](#)  
*Allow use as an STL container.*
- typedef [MSetIterator](#) [iterator](#)  
*Allow use as an STL container.*
- typedef [MSetIterator](#) [const\\_iterator](#)  
*Allow use as an STL container.*
- typedef [MSetIterator](#) & [reference](#)  
*Allow use as an STL container.*
- typedef [MSetIterator](#) & [const\\_reference](#)  
*Allow use as an STL container.*
- typedef [MSetIterator](#) \* [pointer](#)  
*Allow use as an STL container.*
- typedef [Xapian::doccount\\_diff](#) [difference\\_type](#)  
*Allow use as an STL container.*
- typedef [Xapian::doccount](#) [size\\_type](#)  
*Allow use as an STL container.*

### Public Member Functions

- **MSet** ([MSet::Internal](#) \*internal\_)
- [MSet](#) ()  
*Create an empty [Xapian::MSet](#).*
- [~MSet](#) ()  
*Destroy a [Xapian::MSet](#).*
- [MSet](#) (const [MSet](#) &other)  
*Copying is allowed (and is cheap).*
- void [operator=](#) (const [MSet](#) &other)  
*Assignment is allowed (and is cheap).*
- void [fetch](#) (const [MSetIterator](#) &begin, const [MSetIterator](#) &end) const

*Fetch the document info for a set of items in the [MSet](#).*

- void [fetch](#) (const [MSetIterator](#) &item) const  
*Fetch the single item specified.*
- void [fetch](#) () const  
*Fetch all the items in the [MSet](#).*
- [Xapian::percent convert\\_to\\_percent](#) ([Xapian::weight](#) wt) const  
*This converts the weight supplied to a percentage score.*
- [Xapian::percent convert\\_to\\_percent](#) (const [MSetIterator](#) &it) const  
*Return the percentage score for a particular item.*
- [Xapian::doccount get\\_termfreq](#) (const std::string &tname) const  
*Return the term frequency of the given query term.*
- [Xapian::weight get\\_termweight](#) (const std::string &tname) const  
*Return the term weight of the given query term.*
- [Xapian::doccount get\\_firstitem](#) () const  
*The index of the first item in the result which was put into the [MSet](#).*
- [Xapian::doccount get\\_matches\\_lower\\_bound](#) () const  
*A lower bound on the number of documents in the database which match the query.*
- [Xapian::doccount get\\_matches\\_estimated](#) () const  
*An estimate for the number of documents in the database which match the query.*
- [Xapian::doccount get\\_matches\\_upper\\_bound](#) () const  
*An upper bound on the number of documents in the database which match the query.*
- [Xapian::weight get\\_max\\_possible](#) () const  
*The maximum possible weight in the [MSet](#).*
- [Xapian::weight get\\_max\\_attained](#) () const  
*The greatest weight which is attained by any document in the database.*
- [Xapian::doccount size](#) () const  
*The number of items in this [MSet](#).*
- [Xapian::doccount max\\_size](#) () const  
*Required to allow use as an STL container.*
- bool [empty](#) () const  
*Test if this [MSet](#) is empty.*

- void `swap (MSet &other)`  
*Swap the `MSet` we point to with another.*
- `MSetIterator begin () const`  
*Iterator for the terms in this `MSet`.*
- `MSetIterator end () const`  
*End iterator corresponding to `begin()`.*
- `MSetIterator back () const`  
*Iterator pointing to the last element of this `MSet`.*
- `MSetIterator operator[] (Xapian::doccount i) const`  
*This returns the document at position `i` in this `MSet` object.*
- `std::string get_description () const`  
*Return a string describing this object.*

## Public Attributes

- `Xapian::Internal::RefCntPtr< Internal > internal`

## 7.29.1 Detailed Description

A match set (`MSet`).

This class represents (a portion of) the results of a query.

## 7.29.2 Member Typedef Documentation

### 7.29.2.1 `typedef MSetIterator Xapian::MSet::const_iterator`

Allow use as an STL container.

### 7.29.2.2 `typedef MSetIterator& Xapian::MSet::const_reference`

Allow use as an STL container.

### 7.29.2.3 `typedef Xapian::doccount_diff Xapian::MSet::difference_type`

Allow use as an STL container.



**7.29.2.4 typedef MSetIterator Xapian::MSet::iterator**

Allow use as an STL container.

**7.29.2.5 typedef MSetIterator\* Xapian::MSet::pointer**

Allow use as an STL container.

**7.29.2.6 typedef MSetIterator& Xapian::MSet::reference**

Allow use as an STL container.

**7.29.2.7 typedef Xapian::doccount Xapian::MSet::size\_type**

Allow use as an STL container.

**7.29.2.8 typedef MSetIterator Xapian::MSet::value\_type**

Allow use as an STL container.

**7.29.3 Constructor & Destructor Documentation****7.29.3.1 Xapian::MSet::MSet ()**

Create an empty [Xapian::MSet](#).

**7.29.3.2 Xapian::MSet::~~MSet ()**

Destroy a [Xapian::MSet](#).

**7.29.3.3 Xapian::MSet::MSet (const MSet & *other*)**

Copying is allowed (and is cheap).

**7.29.4 Member Function Documentation****7.29.4.1 MSetIterator Xapian::MSet::back () const**

Iterator pointing to the last element of this [MSet](#).

**7.29.4.2 MSetIterator Xapian::MSet::begin () const**

Iterator for the terms in this [MSet](#).

#### 7.29.4.3 **Xapian::percent Xapian::MSet::convert\_to\_percent (const MSetIterator & *it*) const**

Return the percentage score for a particular item.

#### 7.29.4.4 **Xapian::percent Xapian::MSet::convert\_to\_percent (Xapian::weight *wt*) const**

This converts the weight supplied to a percentage score.

The return value will be in the range 0 to 100, and will be 0 if and only if the item did not match the query at all.

#### 7.29.4.5 **bool Xapian::MSet::empty () const**

Test if this [MSet](#) is empty.

#### 7.29.4.6 **MSetIterator Xapian::MSet::end () const**

End iterator corresponding to [begin\(\)](#).

#### 7.29.4.7 **void Xapian::MSet::fetch () const**

Fetch all the items in the [MSet](#).

#### 7.29.4.8 **void Xapian::MSet::fetch (const MSetIterator & *item*) const**

Fetch the single item specified.

#### 7.29.4.9 **void Xapian::MSet::fetch (const MSetIterator & *begin*, const MSetIterator & *end*) const**

Fetch the document info for a set of items in the [MSet](#).

This method causes the documents in the range specified by the iterators to be fetched from the database, and cached in the [Xapian::MSet](#) object. This has little effect when performing a search across a local database, but will greatly speed up subsequent access to the document contents when the documents are stored in a remote database.

The iterators must be over this [Xapian::MSet](#) - undefined behaviour will result otherwise.

#### **Parameters:**

*begin* [MSetIterator](#) for first item to fetch.

*end* [MSetIterator](#) for item after last item to fetch.

**7.29.4.10 `std::string Xapian::MSet::get_description () const`**

Return a string describing this object.

**7.29.4.11 `Xapian::doccount Xapian::MSet::get_firstitem () const`**

The index of the first item in the result which was put into the [MSet](#).

This corresponds to the parameter "first" specified in [Xapian::Enquire::get\\_mset\(\)](#). A value of 0 corresponds to the highest result being the first item in the [MSet](#).

**7.29.4.12 `Xapian::doccount Xapian::MSet::get_matches_estimated () const`**

An estimate for the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This value is returned because there is sometimes a request to display such information. However, our experience is that presenting this value to users causes them to worry about the large number of results, rather than how useful those at the top of the result set are, and is thus undesirable.

**7.29.4.13 `Xapian::doccount Xapian::MSet::get_matches_lower_bound () const`**

A lower bound on the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This number is usually considerably less than the actual number of documents which match the query.

**7.29.4.14 `Xapian::doccount Xapian::MSet::get_matches_upper_bound () const`**

An upper bound on the number of documents in the database which match the query.

This figure takes into account collapsing of duplicates, and weighting cutoff values.

This number is usually considerably greater than the actual number of documents which match the query.

**7.29.4.15 `Xapian::weight Xapian::MSet::get_max_attained () const`**

The greatest weight which is attained by any document in the database.

If `firstitem == 0` and the primary ordering is by relevance, this is the weight of the first entry in the [MSet](#).

If no documents are found by the query, this will be 0.

Note that calculation of `max_attained` requires calculation of at least one result item - therefore, if no items were requested when the query was performed (by specifying

maxitems = 0 in [Xapian::Enquire::get\\_mset\(\)](#), this value will be 0.

#### 7.29.4.16 **Xapian::weight Xapian::MSet::get\_max\_possible () const**

The maximum possible weight in the [MSet](#).

This weight is likely not to be attained in the set of results, but represents an upper bound on the weight which a document could attain for the given query.

#### 7.29.4.17 **Xapian::doccount Xapian::MSet::get\_termfreq (const std::string & tname) const**

Return the term frequency of the given query term.

##### **Parameters:**

*tname* The term to look for.

##### **Exceptions:**

[Xapian::InvalidArgumentError](#) is thrown if the term was not in the query.

#### 7.29.4.18 **Xapian::weight Xapian::MSet::get\_termweight (const std::string & tname) const**

Return the term weight of the given query term.

##### **Parameters:**

*tname* The term to look for.

##### **Exceptions:**

[Xapian::InvalidArgumentError](#) is thrown if the term was not in the query.

#### 7.29.4.19 **Xapian::doccount Xapian::MSet::max\_size () const** `[inline]`

Required to allow use as an STL container.

#### 7.29.4.20 **void Xapian::MSet::operator= (const MSet & other)**

Assignment is allowed (and is cheap).

**7.29.4.21 MSetIterator Xapian::MSet::operator[] (Xapian::doccount *i*) const**

This returns the document at position *i* in this [MSet](#) object.

Note that this is not the same as the document at rank *i* in the query, unless the "first" parameter to [Xapian::Enquire::get\\_mset](#) was 0. Rather, it is the document at rank *i* + first.

In other words, the offset is into the documents represented by this object, not into the set of documents matching the query.

**7.29.4.22 Xapian::doccount Xapian::MSet::size () const**

The number of items in this [MSet](#).

**7.29.4.23 void Xapian::MSet::swap (MSet & *other*)**

Swap the [MSet](#) we point to with another.

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

- [xapian/enquire.h](#)

## 7.30 Xapian::MSetIterator Class Reference

An iterator pointing to items in an [MSet](#).

### Public Types

- typedef std::bidirectional\_iterator\_tag [iterator\\_category](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::docid](#) value\_type  
*Allow use as an STL iterator.*
- typedef [Xapian::doccount\\_diff](#) difference\_type  
*Allow use as an STL iterator.*
- typedef [Xapian::docid](#) \* pointer  
*Allow use as an STL iterator.*
- typedef [Xapian::docid](#) & reference  
*Allow use as an STL iterator.*

### Public Member Functions

- [MSetIterator](#) ()  
*Create an uninitialised iterator; this cannot be used, but is convenient syntactically.*
- [MSetIterator](#) (const [MSetIterator](#) &other)  
*Copying is allowed (and is cheap).*
- void operator= (const [MSetIterator](#) &other)  
*Assignment is allowed (and is cheap).*
- [MSetIterator](#) & operator++ ()  
*Advance the iterator.*
- [MSetIterator](#) operator++ (int)  
*Advance the iterator (postfix variant).*
- [MSetIterator](#) & operator-- ()  
*Decrement the iterator.*
- [MSetIterator](#) operator-- (int)  
*Decrement the iterator (postfix variant).*
- [Xapian::docid](#) operator\* () const

*Get the document ID for the current position.*

- [Xapian::Document get\\_document \(\)](#) const  
*Get a [Xapian::Document](#) object for the current position.*
- [Xapian::doccount get\\_rank \(\)](#) const  
*Get the rank of the document at the current position.*
- [Xapian::weight get\\_weight \(\)](#) const  
*Get the weight of the document at the current position.*
- [std::string get\\_collapse\\_key \(\)](#) const  
*Get the collapse key for this document.*
- [Xapian::doccount get\\_collapse\\_count \(\)](#) const  
*Get an estimate of the number of documents that have been collapsed into this one.*
- [Xapian::percent get\\_percent \(\)](#) const  
*This returns the weight of the document as a percentage score.*
- [std::string get\\_description \(\)](#) const  
*Return a string describing this object.*

## Friends

- [bool operator== \(const \[MSetIterator\]\(#\) &a, const \[MSetIterator\]\(#\) &b\)](#)  
*Equality test for [MSetIterator](#) objects.*
- [bool operator!= \(const \[MSetIterator\]\(#\) &a, const \[MSetIterator\]\(#\) &b\)](#)  
*Inequality test for [MSetIterator](#) objects.*

### 7.30.1 Detailed Description

An iterator pointing to items in an [MSet](#).

This is used for access to individual results of a match.

### 7.30.2 Member Typedef Documentation

#### 7.30.2.1 `typedef Xapian::doccount_diff Xapian::MSetIterator::difference_type`

Allow use as an STL iterator.

### 7.30.2.2 `typedef std::bidirectional_iterator_tag` `Xapian::MSetIterator::iterator_category`

Allow use as an STL iterator.

### 7.30.2.3 `typedef Xapian::docid* Xapian::MSetIterator::pointer`

Allow use as an STL iterator.

### 7.30.2.4 `typedef Xapian::docid& Xapian::MSetIterator::reference`

Allow use as an STL iterator.

### 7.30.2.5 `typedef Xapian::docid Xapian::MSetIterator::value_type`

Allow use as an STL iterator.

## 7.30.3 Constructor & Destructor Documentation

### 7.30.3.1 `Xapian::MSetIterator::MSetIterator ()` `[inline]`

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

### 7.30.3.2 `Xapian::MSetIterator::MSetIterator (const MSetIterator & other)` `[inline]`

Copying is allowed (and is cheap).

## 7.30.4 Member Function Documentation

### 7.30.4.1 `Xapian::doccount Xapian::MSetIterator::get_collapse_count () const`

Get an estimate of the number of documents that have been collapsed into this one.

The estimate will always be less than or equal to the actual number of other documents satisfying the match criteria with the same collapse key as this document.

This method may return 0 even though there are other documents with the same collapse key which satisfying the match criteria. However if this method returns non-zero, there definitely are other such documents. So this method may be used to inform the user that there are "at least N other matches in this group", or to control whether to offer a "show other documents in this group" feature (but note that it may not offer it in every case where it would show other documents).



#### 7.30.4.2 `std::string Xapian::MSetIterator::get_collapse_key () const`

Get the collapse key for this document.

#### 7.30.4.3 `std::string Xapian::MSetIterator::get_description () const`

Return a string describing this object.

#### 7.30.4.4 `Xapian::Document Xapian::MSetIterator::get_document () const`

Get a [Xapian::Document](#) object for the current position.

This method returns a [Xapian::Document](#) object which provides the information about the document pointed to by the [MSetIterator](#).

If the underlying database has suitable support, using this call (rather than asking the database for a document based on its document ID) will enable the system to ensure that the correct data is returned, and that the document has not been deleted or changed since the query was performed.

##### Returns:

A [Xapian::Document](#) object containing the document data.

##### Exceptions:

[Xapian::DocNotFoundError](#) The document specified could not be found in the database.

#### 7.30.4.5 `Xapian::percent Xapian::MSetIterator::get_percent () const`

This returns the weight of the document as a percentage score.

The return value will be an integer in the range 0 to 100: 0 meaning that the item did not match the query at all.

The intention is that the highest weighted document will get 100 if it matches all the weight-contributing terms in the query. However, currently it may get a lower percentage score if you use a [MatchDecider](#) and the sorting is primarily by value. In this case, the percentage for a particular document may vary depending on the first, max\_size, and checkatleast parameters passed to [Enquire::get\\_mset\(\)](#) (this bug is hard to fix without having to apply the [MatchDecider](#) to potentially many more documents, which is potentially costly).

#### 7.30.4.6 `Xapian::doccount Xapian::MSetIterator::get_rank () const` `[inline]`

Get the rank of the document at the current position.

The rank is the position that this document is at in the ordered list of results of the query. The result is 0-based - i.e. the top-ranked document has a rank of 0.

#### 7.30.4.7 **Xapian::weight Xapian::MSetIterator::get\_weight () const**

Get the weight of the document at the current position.

#### 7.30.4.8 **Xapian::docid Xapian::MSetIterator::operator\* () const**

Get the document ID for the current position.

#### 7.30.4.9 **MSetIterator Xapian::MSetIterator::operator++ (int) [inline]**

Advance the iterator (postfix variant).

#### 7.30.4.10 **MSetIterator& Xapian::MSetIterator::operator++ () [inline]**

Advance the iterator.

#### 7.30.4.11 **MSetIterator Xapian::MSetIterator::operator-- (int) [inline]**

Decrement the iterator (postfix variant).

#### 7.30.4.12 **MSetIterator& Xapian::MSetIterator::operator-- () [inline]**

Decrement the iterator.

#### 7.30.4.13 **void Xapian::MSetIterator::operator= (const MSetIterator & *other*) [inline]**

Assignment is allowed (and is cheap).

### 7.30.5 Friends And Related Function Documentation

#### 7.30.5.1 **bool operator!= (const MSetIterator & *a*, const MSetIterator & *b*) [friend]**

Inequality test for [MSetIterator](#) objects.

#### 7.30.5.2 **bool operator== (const MSetIterator & *a*, const MSetIterator & *b*) [friend]**

Equality test for [MSetIterator](#) objects.

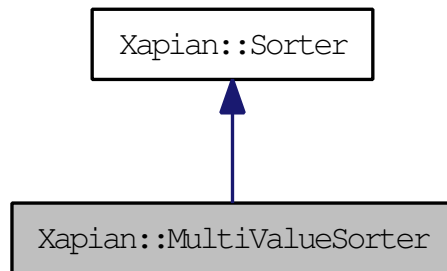
The documentation for this class was generated from the following file:

- [xapian/enquire.h](#)

## 7.31 Xapian::MultiValueSorter Class Reference

[Sorter](#) subclass which sorts by a several values.

Inheritance diagram for Xapian::MultiValueSorter:



### Public Member Functions

- `template<class Iterator >`  
**MultiValueSorter** (Iterator begin, Iterator end)
- `virtual std::string operator() (const Xapian::Document &doc) const`  
*This method takes a [Document](#) object and builds a sort key from it.*
- `void add (Xapian::valueno valno, bool forward=true)`

#### 7.31.1 Detailed Description

[Sorter](#) subclass which sorts by a several values.

Results are ordered by the first value. In the event of a tie, the second is used. If this is the same for both, the third is used, and so on.

#### 7.31.2 Member Function Documentation

##### 7.31.2.1 `virtual std::string Xapian::MultiValueSorter::operator() (const Xapian::Document & doc) const` [virtual]

This method takes a [Document](#) object and builds a sort key from it.

Documents are then ordered by a string compare on the sort keys.

Implements [Xapian::Sorter](#).

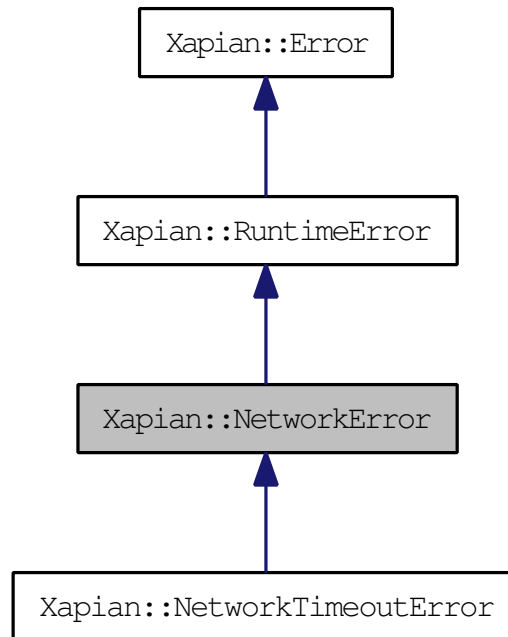
The documentation for this class was generated from the following file:

- `xapian/sorter.h`

## 7.32 Xapian::NetworkError Class Reference

Indicates a problem communicating with a remote database.

Inheritance diagram for Xapian::NetworkError:



### Public Member Functions

- [NetworkError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [NetworkError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.32.1 Detailed Description

Indicates a problem communicating with a remote database.

## 7.32.2 Constructor & Destructor Documentation

**7.32.2.1 Xapian::NetworkError::NetworkError** (const std::string & *msg\_*,  
const std::string & *context\_* = "", int *errno\_* = 0) [inline,  
explicit]

General purpose constructor which allows setting errno.

**7.32.2.2 Xapian::NetworkError::NetworkError** (const std::string & *msg\_*, int  
*errno\_*) [inline]

Construct from message and errno value.

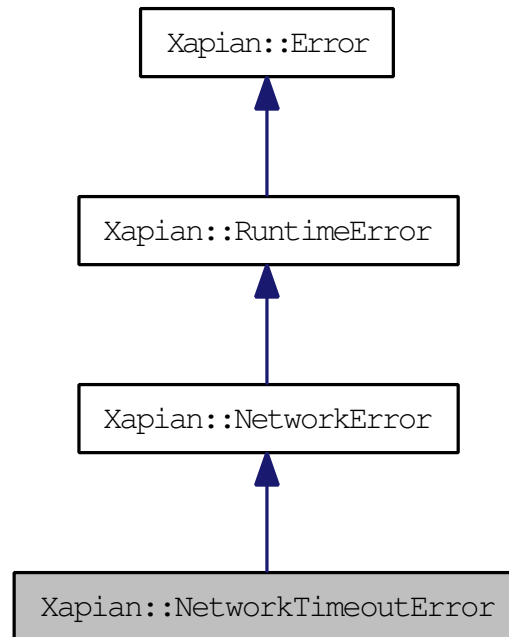
The documentation for this class was generated from the following file:

- xapian/[error.h](#)

## 7.33 Xapian::NetworkTimeoutError Class Reference

Indicates a timeout expired while communicating with a remote database.

Inheritance diagram for Xapian::NetworkTimeoutError:



### Public Member Functions

- [NetworkTimeoutError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [NetworkTimeoutError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.33.1 Detailed Description

Indicates a timeout expired while communicating with a remote database.

## 7.33.2 Constructor & Destructor Documentation

### 7.33.2.1 Xapian::NetworkTimeoutError::NetworkTimeoutError (const std::string & *msg\_*, const std::string & *context\_* = "", int *errno\_* = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.33.2.2 Xapian::NetworkTimeoutError::NetworkTimeoutError (const std::string & *msg\_*, int *errno\_*) [inline]

Construct from message and errno value.

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

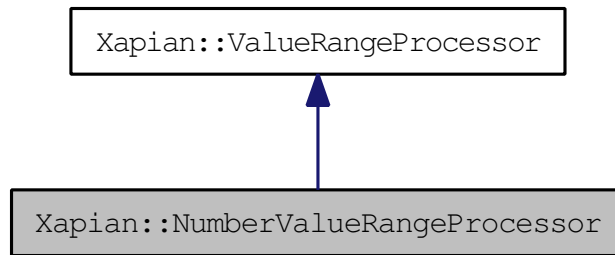
- xapian/[error.h](#)



## 7.34 Xapian::NumberValueRangeProcessor Class Reference

Handle a number range.

Inheritance diagram for Xapian::NumberValueRangeProcessor:



### Public Member Functions

- [NumberValueRangeProcessor](#) ([Xapian::valueno](#) valno\_)  
*Constructor.*
- [NumberValueRangeProcessor](#) ([Xapian::valueno](#) valno\_, const std::string &str\_, bool prefix\_=true)  
*Constructor.*
- [Xapian::valueno operator\(\)](#) (std::string &begin, std::string &end)  
*Check for a valid range of this type.*

#### 7.34.1 Detailed Description

Handle a number range.

This class must be used on values which have been encoded using [Xapian::sortable\\_serialise\(\)](#) which turns numbers into strings which will sort in the same order as the numbers (the same values can be used to implement a numeric sort).

#### 7.34.2 Constructor & Destructor Documentation

##### 7.34.2.1 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno valno\_) [inline]

Constructor.

**Parameters:**

*valno\_* The value number to return from operator().

#### 7.34.2.2 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *valno\_*, const std::string & *str\_*, bool *prefix\_* = true) [inline]

Constructor.

**Parameters:**

*valno\_* The value number to return from operator().

*str\_* A string to look for to recognise values as belonging to this numeric range.

*prefix\_* Whether to look for the string at the start or end of the values. If true, the string is a prefix; if false, the string is a suffix (default: true).

The string supplied in *str\_* is used by *operator()* to decide whether the pair of strings supplied to it constitute a valid range. If *prefix\_* is true, the first value in a range must begin with *str\_* (and the second value may optionally begin with *str\_*); if *prefix\_* is false, the second value in a range must end with *str\_* (and the first value may optionally end with *str\_*).

If *str\_* is empty, the setting of *prefix\_* is irrelevant, and no special strings are required at the start or end of the strings defining the range.

The remainder of both strings defining the endpoints must be valid floating point numbers. (FIXME: define format recognised).

For example, if *str\_* is "\$" and *prefix\_* is true, and the range processor has been added to the queryparser, the queryparser will accept "\$10..50" or "\$10..\$50", but not "10..50" or "10..\$50" as valid ranges. If *str\_* is "kg" and *prefix\_* is false, the queryparser will accept "10..50kg" or "10kg..50kg", but not "10..50" or "10kg..50" as valid ranges.

### 7.34.3 Member Function Documentation

#### 7.34.3.1 Xapian::valueno Xapian::NumberValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) [virtual]

Check for a valid range of this type.

If BEGIN..END is a valid numeric value range, and has the appropriate prefix or suffix (if specified) required for this [NumberValueRangeProcessor](#), this method returns the value number of range filter on, and sets begin and end to the appropriate serialised values needed to delimit the range. Otherwise it returns [Xapian::BAD\\_VALUENO](#).

Implements [Xapian::ValueRangeProcessor](#).

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

- [xapian/queryparser.h](#)

## 7.35 Xapian::PositionIterator Class Reference

An iterator pointing to items in a list of positions.

### Public Types

- typedef std::input\_iterator\_tag **iterator\_category**
- typedef [Xapian::termpos](#) **value\_type**
- typedef [Xapian::termpos\\_diff](#) **difference\_type**
- typedef [Xapian::termpos](#) \* **pointer**
- typedef [Xapian::termpos](#) & **reference**

### Public Member Functions

- **PositionIterator** (Internal \*internal\_)
- [PositionIterator](#) ()  
*Default constructor - for declaring an uninitialised iterator.*
- [~PositionIterator](#) ()  
*Destructor.*
- [PositionIterator](#) (const [PositionIterator](#) &o)  
*Copying is allowed.*
- void **operator=** (const [PositionIterator](#) &o)  
*Assignment is allowed.*
- [Xapian::termpos](#) **operator\*** () const
- [PositionIterator](#) & **operator++** ()
- TermPosWrapper **operator++** (int)
- void **skip\_to** ([Xapian::termpos](#) pos)  
*Advance the iterator to the specified termpos.*
- std::string **get\_description** () const  
*Return a string describing this object.*

### Friends

- bool **operator==** (const [PositionIterator](#) &a, const [PositionIterator](#) &b)  
*Test equality of two PositionIterators.*

### 7.35.1 Detailed Description

An iterator pointing to items in a list of positions.

### 7.35.2 Constructor & Destructor Documentation

#### 7.35.2.1 Xapian::PositionIterator::PositionIterator ()

Default constructor - for declaring an uninitialised iterator.

#### 7.35.2.2 Xapian::PositionIterator::~~PositionIterator ()

Destructor.

#### 7.35.2.3 Xapian::PositionIterator::PositionIterator (const PositionIterator & o)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

### 7.35.3 Member Function Documentation

#### 7.35.3.1 std::string Xapian::PositionIterator::get\_description () const

Return a string describing this object.

#### 7.35.3.2 void Xapian::PositionIterator::operator= (const PositionIterator & o)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

#### 7.35.3.3 void Xapian::PositionIterator::skip\_to (Xapian::termpos *pos*)

Advance the iterator to the specified termpos.

If the specified termpos isn't in the list, position ourselves on the first termpos after it (or `at_end()` if no greater term positions are present).

### 7.35.4 Friends And Related Function Documentation

#### 7.35.4.1 bool operator== (const PositionIterator & *a*, const PositionIterator & *b*) [`friend`]

Test equality of two PositionIterators.

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

- xapian/[positioniterator.h](#)

## 7.36 Xapian::PostingIterator Class Reference

An iterator pointing to items in a list of postings.

### Public Types

- typedef std::input\_iterator\_tag [iterator\\_category](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::docid](#) [value\\_type](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::doccount\\_diff](#) [difference\\_type](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::docid](#) \* [pointer](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::docid](#) & [reference](#)  
*Allow use as an STL iterator.*

### Public Member Functions

- [PostingIterator](#) ()  
*Default constructor - for declaring an uninitialised iterator.*
- [~PostingIterator](#) ()  
*Destructor.*
- [PostingIterator](#) (const [PostingIterator](#) &other)  
*Copying is allowed.*
- void [operator=](#) (const [PostingIterator](#) &other)  
*Assignment is allowed.*
- [PostingIterator](#) & [operator++](#) ()
- DocIDWrapper [operator++](#) (int)
- void [skip\\_to](#) ([Xapian::docid](#) did)  
*Advance the iterator to the specified docid.*
- [Xapian::docid](#) [operator\\*](#) () const  
*Get the document id at the current position in the postlist.*
- [Xapian::doclength](#) [get\\_doclength](#) () const  
*Get the length of the document at the current position in the postlist.*

- [Xapian::termcount get\\_wdf \(\)](#) const  
*Get the within document frequency of the document at the current position in the postlist.*
- [PositionIterator positionlist\\_begin \(\)](#) const  
*Return [PositionIterator](#) pointing to start of positionlist for current document.*
- [PositionIterator positionlist\\_end \(\)](#) const  
*Return [PositionIterator](#) pointing to end of positionlist for current document.*
- `std::string` [get\\_description \(\)](#) const  
*Return a string describing this object.*

## Friends

- `bool` [operator==](#) (const [PostingIterator](#) &a, const [PostingIterator](#) &b)  
*Test equality of two PostingIterators.*

### 7.36.1 Detailed Description

An iterator pointing to items in a list of postings.

### 7.36.2 Member Typedef Documentation

#### 7.36.2.1 `typedef Xapian::doccount_diff Xapian::PostingIterator::difference_type`

Allow use as an STL iterator.

#### 7.36.2.2 `typedef std::input_iterator_tag Xapian::PostingIterator::iterator_category`

Allow use as an STL iterator.

#### 7.36.2.3 `typedef Xapian::docid* Xapian::PostingIterator::pointer`

Allow use as an STL iterator.

#### 7.36.2.4 `typedef Xapian::docid& Xapian::PostingIterator::reference`

Allow use as an STL iterator.

### 7.36.2.5 `typedef Xapian::docid Xapian::PostingIterator::value_type`

Allow use as an STL iterator.

## 7.36.3 Constructor & Destructor Documentation

### 7.36.3.1 `Xapian::PostingIterator::PostingIterator ()`

Default constructor - for declaring an uninitialised iterator.

### 7.36.3.2 `Xapian::PostingIterator::~~PostingIterator ()`

Destructor.

### 7.36.3.3 `Xapian::PostingIterator::PostingIterator (const PostingIterator & other)`

Copying is allowed.

The internals are reference counted, so copying is also cheap.

## 7.36.4 Member Function Documentation

### 7.36.4.1 `std::string Xapian::PostingIterator::get_description () const`

Return a string describing this object.

### 7.36.4.2 `Xapian::doclength Xapian::PostingIterator::get_doclength () const`

Get the length of the document at the current position in the postlist.

This information may be stored in the postlist, in which case this lookup should be extremely fast (indeed, not require further disk access). If the information is not present in the postlist, it will be retrieved from the database, at a greater performance cost.

### 7.36.4.3 `Xapian::termcount Xapian::PostingIterator::get_wdf () const`

Get the within document frequency of the document at the current position in the postlist.

### 7.36.4.4 `Xapian::docid Xapian::PostingIterator::operator* () const`

Get the document id at the current position in the postlist.



**7.36.4.5 void Xapian::PostingIterator::operator= (const PostingIterator & *other*)**

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

**7.36.4.6 PositionIterator Xapian::PostingIterator::positionlist\_begin () const**

Return [PositionIterator](#) pointing to start of positionlist for current document.

**7.36.4.7 PositionIterator Xapian::PostingIterator::positionlist\_end () const**  
[inline]

Return [PositionIterator](#) pointing to end of positionlist for current document.

**7.36.4.8 void Xapian::PostingIterator::skip\_to (Xapian::docid *did*)**

Advance the iterator to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or at\_end() if no greater docids are present).

**7.36.5 Friends And Related Function Documentation****7.36.5.1 bool operator== (const PostingIterator & *a*, const PostingIterator & *b*)**  
[friend]

Test equality of two PostingIterators.

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

- [xapian/postingiterator.h](#)

## 7.37 Xapian::Query Class Reference

Class representing a query.

### Public Types

- enum `op` {  
`OP_AND`, `OP_OR`, `OP_AND_NOT`, `OP_XOR`,  
`OP_AND_MAYBE`, `OP_FILTER`, `OP_NEAR`, `OP_PHRASE`,  
`OP_VALUE_RANGE`, `OP_SCALE_WEIGHT`, `OP_ELITE_SET`, `OP_VALUE_GE`,  
`OP_VALUE_LE` }

*Enum of possible query operations.*

### Public Member Functions

- `Query` (const `Query` &copyme)  
*Copy constructor.*
- `Query` & `operator=` (const `Query` &copyme)  
*Assignment.*
- `Query` ()  
*Default constructor: makes an empty query which matches no documents.*
- `~Query` ()  
*Destructor.*
- `Query` (const std::string &name\_, `Xapian::termcount` wqf\_=1, `Xapian::termpos` pos\_=0)  
*A query consisting of a single term.*
- `Query` (`Query::op` op\_, const `Query` &left, const `Query` &right)  
*A query consisting of two subqueries, opp-ed together.*
- `Query` (`Query::op` op\_, const std::string &left, const std::string &right)  
*A query consisting of two termnames opp-ed together.*
- template<class Iterator >  
`Query` (`Query::op` op\_, Iterator qbegin, Iterator qend, `Xapian::termcount` parameter=0)  
*Combine a number of `Xapian::Query`-s with the specified operator.*
- `Query` (`Query::op` op\_, `Xapian::Query` q)

Apply the specified operator to a single [Xapian::Query](#) object.

- [Query](#) ([Query::op](#) op\_, [Xapian::Query](#) q, double parameter)  
*Apply the specified operator to a single [Xapian::Query](#) object, with a double parameter.*
- [Query](#) ([Query::op](#) op\_, [Xapian::valueno](#) valno, const std::string &begin, const std::string &end)  
*Construct a value range query on a document value.*
- [Query](#) ([Query::op](#) op\_, [Xapian::valueno](#) valno, const std::string &value)  
*Construct a value comparison query on a document value.*
- [Xapian::termcount](#) [get\\_length](#) () const  
*Get the length of the query, used by some ranking formulae.*
- [TermIterator](#) [get\\_terms\\_begin](#) () const  
*Return a [Xapian::TermIterator](#) returning all the terms in the query, in order of termpos.*
- [TermIterator](#) [get\\_terms\\_end](#) () const  
*Return a [Xapian::TermIterator](#) to the end of the list of terms in the query.*
- bool [empty](#) () const  
*Test if the query is empty (i.e.*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

## Static Public Attributes

- static [Xapian::Query](#) [MatchAll](#)  
*A query which matches all documents in the database.*
- static [Xapian::Query](#) [MatchNothing](#)  
*A query which matches no documents.*

### 7.37.1 Detailed Description

Class representing a query.

Queries are represented as a tree of objects.

## 7.37.2 Member Enumeration Documentation

### 7.37.2.1 enum Xapian::Query::op

Enum of possible query operations.

#### Enumerator:

**OP\_AND** Return iff both subqueries are satisfied.

**OP\_OR** Return if either subquery is satisfied.

**OP\_AND\_NOT** Return if left but not right satisfied.

**OP\_XOR** Return if one query satisfied, but not both.

**OP\_AND\_MAYBE** Return iff left satisfied, but use weights from both.

**OP\_FILTER** As AND, but use only weights from left subquery.

**OP\_NEAR** Find occurrences of a list of terms with all the terms occurring within a specified window of positions.

Each occurrence of a term must be at a different position, but the order they appear in is irrelevant.

The window parameter should be specified for this operation, but will default to the number of terms in the list.

**OP\_PHRASE** Find occurrences of a list of terms with all the terms occurring within a specified window of positions, and all the terms appearing in the order specified.

Each occurrence of a term must be at a different position.

The window parameter should be specified for this operation, but will default to the number of terms in the list.

**OP\_VALUE\_RANGE** Filter by a range test on a document value.

**OP\_SCALE\_WEIGHT** Scale the weight of a subquery by the specified factor.

A factor of 0 means this subquery will contribute no weight to the query - it will act as a purely boolean subquery.

If the factor is negative, [Xapian::InvalidArgumentError](#) will be thrown.

**OP\_ELITE\_SET** Select an elite set from the subqueries, and perform a query with these combined as an OR query.

**OP\_VALUE\_GE** Filter by a greater-than-or-equal test on a document value.

**OP\_VALUE\_LE** Filter by a less-than-or-equal test on a document value.

## 7.37.3 Constructor & Destructor Documentation

### 7.37.3.1 Xapian::Query::Query (const Query & *copyme*)

Copy constructor.

**7.37.3.2 Xapian::Query::Query ()**

Default constructor: makes an empty query which matches no documents.

Also useful for defining a [Query](#) object to be assigned to later.

An exception will be thrown if an attempt is made to use an undefined query when building up a composite query.

**7.37.3.3 Xapian::Query::~~Query ()**

Destructor.

**7.37.3.4 Xapian::Query::Query (const std::string & *tname\_*,  
Xapian::termcount *wqf\_* = 1, Xapian::termpos *pos\_* = 0)**

A query consisting of a single term.

**7.37.3.5 Xapian::Query::Query (Query::op *op\_*, const Query & *left*, const  
Query & *right*)**

A query consisting of two subqueries, opp-ed together.

**7.37.3.6 Xapian::Query::Query (Query::op *op\_*, const std::string & *left*, const  
std::string & *right*)**

A query consisting of two termnames opp-ed together.

**7.37.3.7 template<class Iterator > Xapian::Query::Query (Query::op *op\_*,  
Iterator *qbegin*, Iterator *qend*, Xapian::termcount *parameter* = 0)  
[inline]**

Combine a number of [Xapian::Query](#)-s with the specified operator.

The [Xapian::Query](#) objects are specified with begin and end iterators.

AND, OR, NEAR and PHRASE can take any number of subqueries. Other operators take exactly two subqueries.

The iterators may be to [Xapian::Query](#) objects, pointers to [Xapian::Query](#) objects, or termnames (std::string-s).

For NEAR and PHRASE, a window size can be specified in parameter.

For ELITE\_SET, the elite set size can be specified in parameter.

**7.37.3.8 Xapian::Query::Query (Query::op *op\_*, Xapian::Query *q*)**

Apply the specified operator to a single [Xapian::Query](#) object.

### Deprecated

This method is deprecated because it isn't useful, since none of the current query operators can be usefully applied to a single subquery without a parameter value.

#### 7.37.3.9 **Xapian::Query::Query (Query::op *op\_*, Xapian::Query *q*, double *parameter*)**

Apply the specified operator to a single [Xapian::Query](#) object, with a double parameter.

#### 7.37.3.10 **Xapian::Query::Query (Query::op *op\_*, Xapian::valueno *valno*, const std::string & *begin*, const std::string & *end*)**

Construct a value range query on a document value.

A value range query matches those documents which have a value stored in the slot given by *valno* which is in the range specified by *begin* and *end* (in lexicographical order), including the endpoints.

##### Parameters:

*op\_* The operator to use for the query. Currently, must be OP\_VALUE\_RANGE.

*valno* The slot number to get the value from.

*begin* The start of the range.

*end* The end of the range.

#### 7.37.3.11 **Xapian::Query::Query (Query::op *op\_*, Xapian::valueno *valno*, const std::string & *value*)**

Construct a value comparison query on a document value.

This query matches those documents which have a value stored in the slot given by *valno* which compares, as specified by the operator, to *value*.

##### Parameters:

*op\_* The operator to use for the query. Currently, must be OP\_VALUE\_GE or OP\_VALUE\_LE.

*valno* The slot number to get the value from.

*value* The value to compare.

### 7.37.4 Member Function Documentation

#### 7.37.4.1 **bool Xapian::Query::empty () const**

Test if the query is empty (i.e.

was constructed using the default ctor or with an empty iterator ctor).

#### 7.37.4.2 `std::string Xapian::Query::get_description () const`

Return a string describing this object.

#### 7.37.4.3 `Xapian::termcount Xapian::Query::get_length () const`

Get the length of the query, used by some ranking formulae.

This value is calculated automatically - if you want to override it you can pass a different value to [Enquire::set\\_query\(\)](#).

#### 7.37.4.4 `TermIterator Xapian::Query::get_terms_begin () const`

Return a [Xapian::TermIterator](#) returning all the terms in the query, in order of termpos.

If multiple terms have the same term position, their order is unspecified. Duplicates (same term and termpos) will be removed.

#### 7.37.4.5 `TermIterator Xapian::Query::get_terms_end () const` `[inline]`

Return a [Xapian::TermIterator](#) to the end of the list of terms in the query.

#### 7.37.4.6 `Query& Xapian::Query::operator= (const Query & copyme)`

Assignment.

### 7.37.5 Member Data Documentation

#### 7.37.5.1 `Xapian::Query Xapian::Query::MatchAll` `[static]`

A query which matches all documents in the database.

#### 7.37.5.2 `Xapian::Query Xapian::Query::MatchNothing` `[static]`

A query which matches no documents.

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

- [xapian/query.h](#)

## 7.38 Xapian::QueryParser Class Reference

Build a [Xapian::Query](#) object from a user query string.

### Public Types

- enum [feature\\_flag](#) {  
    [FLAG\\_BOOLEAN](#) = 1, [FLAG\\_PHRASE](#) = 2, [FLAG\\_LOVEHATE](#) = 4,  
    [FLAG\\_BOOLEAN\\_ANY\\_CASE](#) = 8,  
    [FLAG\\_WILDCARD](#) = 16, [FLAG\\_PURE\\_NOT](#) = 32, [FLAG\\_PARTIAL](#) = 64,  
    [FLAG\\_SPELLING\\_CORRECTION](#) = 128,  
    [FLAG\\_SYNONYM](#) = 256, [FLAG\\_AUTO\\_SYNONYMS](#) = 512, [FLAG\\_-](#)  
    [AUTO\\_MULTIWORD\\_SYNONYMS](#) = 1024 | [FLAG\\_AUTO\\_SYNONYMS](#),  
    [FLAG\\_DEFAULT](#) = [FLAG\\_PHRASE](#)|[FLAG\\_BOOLEAN](#)|[FLAG\\_LOVEHATE](#)  
}
- Enum of feature flags.
- enum [stem\\_strategy](#) { [STEM\\_NONE](#), [STEM\\_SOME](#), [STEM\\_ALL](#) }

### Public Member Functions

- [QueryParser](#) (const [QueryParser](#) &o)  
    Copy constructor.
- [QueryParser](#) & operator= (const [QueryParser](#) &o)  
    Assignment.
- [QueryParser](#) ()  
    Default constructor.
- [~QueryParser](#) ()  
    Destructor.
- void [set\\_stemmer](#) (const [Xapian::Stem](#) &stemmer)  
    Set the stemmer.
- void [set\\_stemming\\_strategy](#) (stem\_strategy strategy)  
    Set the stemming strategy.
- void [set\\_stopper](#) (const [Stopper](#) \*stop=NULL)  
    Set the stopper.
- void [set\\_default\\_op](#) ([Query::op](#) default\_op)  
    Set the default operator.



- [Query::op get\\_default\\_op](#) () const  
*Get the current default operator.*
- void [set\\_database](#) (const [Database](#) &db)  
*Specify the database being searched.*
- [Query parse\\_query](#) (const std::string &query\_string, unsigned flags=FLAG\_PHRASE|FLAG\_BOOLEAN|FLAG\_LOVEHATE, const std::string &default\_prefix="")  
*Parse a query.*
- void [add\\_prefix](#) (const std::string &field, const std::string &prefix)  
*Add a probabilistic term prefix.*
- void [add\\_boolean\\_prefix](#) (const std::string &field, const std::string &prefix)  
*Add a boolean term prefix allowing the user to restrict a search with a boolean filter specified in the free text query.*
- [TermIterator stoplist\\_begin](#) () const  
*Iterate over terms omitted from the query as stopwords.*
- [TermIterator stoplist\\_end](#) () const
- [TermIterator unstem\\_begin](#) (const std::string &term) const  
*Iterate over unstemmed forms of the given (stemmed) term used in the query.*
- [TermIterator unstem\\_end](#) (const std::string &) const
- void [add\\_valuerangeprocessor](#) ([Xapian::ValueRangeProcessor](#) \*vrproc)  
*Register a [ValueRangeProcessor](#).*
- std::string [get\\_corrected\\_query\\_string](#) () const  
*Get the spelling-corrected query string.*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

### 7.38.1 Detailed Description

Build a [Xapian::Query](#) object from a user query string.

### 7.38.2 Member Enumeration Documentation

#### 7.38.2.1 enum Xapian::QueryParser::feature\_flag

Enum of feature flags.

**Enumerator:**

**FLAG\_BOOLEAN** Support AND, OR, etc and bracketed subexpressions.

**FLAG\_PHRASE** Support quoted phrases.

**FLAG\_LOVEHATE** Support + and -.

**FLAG\_BOOLEAN\_ANY\_CASE** Support AND, OR, etc even if they aren't in ALLCAPS.

**FLAG\_WILDCARD** Support right truncation (e.g.

Xap\*).

Currently you can't use wildcards with boolean filter prefixes, or in a phrase (either an explicitly quoted one, or one implicitly generated by hyphens or other punctuation).

NB: You need to tell the [QueryParser](#) object which database to expand wildcards from by calling `set_database`.

**FLAG\_PURE\_NOT** Allow queries such as 'NOT apples'.

These require the use of a list of all documents in the database which is potentially expensive, so this feature isn't enabled by default.

**FLAG\_PARTIAL** Enable partial matching.

Partial matching causes the parser to treat the query as a "partially entered" search. This will automatically treat the final word as a wildcarded match, unless it is followed by whitespace, to produce more stable results from interactive searches.

Currently FLAG\_PARTIAL doesn't do anything if the final word in the query has a boolean filter prefix, or if it is in a phrase (either an explicitly quoted one, or one implicitly generated by hyphens or other punctuation). It also doesn't do anything if the final word is part of a value range.

NB: You need to tell the [QueryParser](#) object which database to expand wildcards from by calling `set_database`.

**FLAG\_SPELLING\_CORRECTION** Enable spelling correction.

For each word in the query which doesn't exist as a term in the database, [Database::get\\_spelling\\_suggestion\(\)](#) will be called and if a suggestion is returned, a corrected version of the query string will be built up which can be read using [QueryParser::get\\_corrected\\_query\\_string\(\)](#). The query returned is based on the uncorrected query string however - if you want a parsed query based on the corrected query string, you must call [QueryParser::parse\\_query\(\)](#) again.

NB: You must also call `set_database()` for this to work.

**FLAG\_SYNONYM** Enable synonym operator '~'.

NB: You must also call `set_database()` for this to work.

**FLAG\_AUTO\_SYNONYMS** Enable automatic use of synonyms for single terms.

NB: You must also call `set_database()` for this to work.

**FLAG\_AUTO\_MULTIWORD\_SYNONYMS** Enable automatic use of synonyms for single terms and groups of terms.

NB: You must also call `set_database()` for this to work.

**FLAG\_DEFAULT** The default flags.

Used if you don't explicitly pass any to [parse\\_query\(\)](#).

Added in [Xapian](#) 1.0.11.

### 7.38.3 Constructor & Destructor Documentation

#### 7.38.3.1 Xapian::QueryParser::QueryParser (const QueryParser & o)

Copy constructor.

#### 7.38.3.2 Xapian::QueryParser::QueryParser ()

Default constructor.

#### 7.38.3.3 Xapian::QueryParser::~~QueryParser ()

Destructor.

### 7.38.4 Member Function Documentation

#### 7.38.4.1 void Xapian::QueryParser::add\_boolean\_prefix (const std::string & field, const std::string & prefix)

Add a boolean term prefix allowing the user to restrict a search with a boolean filter specified in the free text query.

For example:

```
qp.add_boolean_prefix("site", "H");
```

This allows the user to restrict a search with `site:xapian.org` which will be converted to `Hxapian.org` combined with any probabilistic query with [Xapian::Query::OP\\_FILTER](#).

If multiple boolean filters are specified in a query for the same prefix, they will be combined with the [Xapian::Query::OP\\_OR](#) operator. Then, if there are boolean filters for different prefixes, they will be combined with the [Xapian::Query::OP\\_AND](#) operator.

Multiple fields can be mapped to the same prefix (so for example you can make `site:` and `domain:` aliases for each other). Instances of fields with different aliases but the same prefix will still be combined with the OR operator.

For example, if `"site"` and `"domain"` map to `"H"`, but `author` maps to `"A"`, a search for `"site:foo domain:bar author:Fred"` will map to `"(Hfoo OR Hbar) AND Afred"`.

As of 1.0.4, you can call this method multiple times with the same value of field to allow a single field to be mapped to multiple prefixes. Multiple terms being generated for such a field, and combined with [Xapian::Query::OP\\_OR](#).

Calling this method with an empty string for *field* will cause a `Xapian::InvalidArgumentError`.

If you call `add_prefix()` and `add_boolean_prefix()` for the same value of *field*, a `Xapian::InvalidOperationError` exception will be thrown.

In 1.0.3 and earlier, subsequent calls to this method with the same value of *field* had no effect.

#### Parameters:

*field* The user visible field name

*prefix* The term prefix to map this to

#### 7.38.4.2 void Xapian::QueryParser::add\_prefix (const std::string &field, const std::string &prefix)

Add a probabilistic term prefix.

For example:

```
qp.add_prefix("author", "A");
```

This allows the user to search for author:Orwell which will be converted to a search for the term "Aorwell".

Multiple fields can be mapped to the same prefix. For example, you can make title: and subject: aliases for each other.

As of 1.0.4, you can call this method multiple times with the same value of *field* to allow a single field to be mapped to multiple prefixes. Multiple terms being generated for such a field, and combined with `Xapian::Query::OP_OR`.

If any prefixes are specified for the empty field name (i.e. you call this method with an empty string as the first parameter) these prefixes will be used for terms without a field specifier. If you do this and also specify the `default_prefix` parameter to `parse_query()`, then the `default_prefix` parameter will override.

If the prefix parameter is empty, then "field:word" will produce the term "word" (and this can be one of several prefixes for a particular field, or for terms without a field specifier).

If you call `add_prefix()` and `add_boolean_prefix()` for the same value of *field*, a `Xapian::InvalidOperationError` exception will be thrown.

In 1.0.3 and earlier, subsequent calls to this method with the same value of *field* had no effect.

#### Parameters:

*field* The user visible field name

*prefix* The term prefix to map this to

#### 7.38.4.3 void Xapian::QueryParser::add\_valuerangeprocessor (Xapian::ValueRangeProcessor \* *vrproc*)

Register a [ValueRangeProcessor](#).

#### 7.38.4.4 std::string Xapian::QueryParser::get\_corrected\_query\_string () const

Get the spelling-corrected query string.

This will only be set if FLAG\_SPELLING\_CORRECTION is specified when [QueryParser::parse\\_query\(\)](#) was last called.

If there were no corrections, an empty string is returned.

#### 7.38.4.5 Query::op Xapian::QueryParser::get\_default\_op () const

Get the current default operator.

#### 7.38.4.6 std::string Xapian::QueryParser::get\_description () const

Return a string describing this object.

#### 7.38.4.7 QueryParser& Xapian::QueryParser::operator= (const QueryParser & *o*)

Assignment.

#### 7.38.4.8 Query Xapian::QueryParser::parse\_query (const std::string & *query\_string*, unsigned *flags* = FLAG\_PHRASE|FLAG\_BOOLEAN|FLAG\_LOVEHATE, const std::string & *default\_prefix* = "")

Parse a query.

##### Parameters:

*query\_string* A free-text query as entered by a user

*flags* Zero or more Query::feature\_flag specifying what features the [Query-Parser](#) should support. Combine multiple values with bitwise-or (|) (default FLAG\_DEFAULT).

*default\_prefix* The default term prefix to use (default none). For example, you can pass "A" when parsing an "Author" field.

##### Exceptions:

*If* the query string can't be parsed, then [Xapian::QueryParserError](#) is thrown. You can get an English error message to report to the user by catching it and

calling `get_msg()` on the caught exception. The current possible values (in case you want to translate them) are:

- Unknown range operation
- parse error
- Syntax: `<expression> AND <expression>`
- Syntax: `<expression> AND NOT <expression>`
- Syntax: `<expression> NOT <expression>`
- Syntax: `<expression> OR <expression>`
- Syntax: `<expression> XOR <expression>`

#### 7.38.4.9 `void Xapian::QueryParser::set_database (const Database & db)`

Specify the database being searched.

The database is used for wildcard expansion (`FLAG_WILDCARD` and `FLAG_PARTIAL`), spelling correction (`FLAG_SPELLING_CORRECTION`), and synonyms (`FLAG_SYNONYM`, `FLAG_AUTO_SYNONYMS`, and `FLAG_AUTO_MULTIWORD_SYNONYMS`).

#### 7.38.4.10 `void Xapian::QueryParser::set_default_op (Query::op default_op)`

Set the default operator.

This operator is used to combine non-filter query items when no explicit operator is used.

The most useful values for this are `OP_OR` (the default) and `OP_AND`. `OP_NEAR` and `OP_PHRASE` can also be useful.

So for example, 'weather forecast' is parsed as if it were 'weather OR forecast' by default.

#### 7.38.4.11 `void Xapian::QueryParser::set_stemmer (const Xapian::Stem & stemmer)`

Set the stemmer.

This sets the stemming algorithm which will be used by the query parser. Note that the stemming algorithm will only be used according to the stemming strategy set by [set\\_stemming\\_strategy\(\)](#), which defaults to `STEM_NONE`. Therefore, to use a stemming algorithm, you will also need to call [set\\_stemming\\_strategy\(\)](#) with a value other than `STEM_NONE`.

**7.38.4.12 void Xapian::QueryParser::set\_stemming\_strategy (stem\_strategy strategy)**

Set the stemming strategy.

This controls how the query parser will apply the stemming algorithm. The default value is STEM\_NONE. The possible values are:

- STEM\_NONE: Don't perform any stemming.
- STEM\_SOME: Search for stemmed forms of terms except for those which start with a capital letter, or are followed by certain characters (currently: (/@<>=\*[{" ), or are used with operators which need positional information. Stemmed terms are prefixed with 'Z'.
- STEM\_ALL: Search for stemmed forms of all words (note: no 'Z' prefix is added).

Note that the stemming algorithm is only applied to words in probabilistic fields - boolean filter terms are never stemmed.

**7.38.4.13 void Xapian::QueryParser::set\_stopper (const Stopper \* stop = NULL)**

Set the stopper.

**7.38.4.14 TermIterator Xapian::QueryParser::stoplist\_begin () const**

Iterate over terms omitted from the query as stopwords.

**7.38.4.15 TermIterator Xapian::QueryParser::unstem\_begin (const std::string & term) const**

Iterate over unstemmed forms of the given (stemmed) term used in the query.

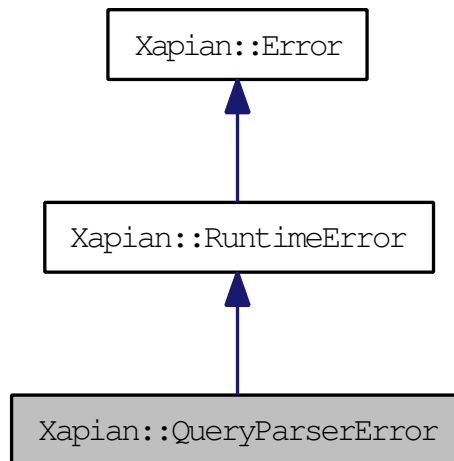
The documentation for this class was generated from the following file:

- [xapian/queryparser.h](#)

## 7.39 Xapian::QueryParserError Class Reference

Indicates a query string can't be parsed.

Inheritance diagram for Xapian::QueryParserError:



### Public Member Functions

- [QueryParserError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [QueryParserError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.39.1 Detailed Description

Indicates a query string can't be parsed.

### 7.39.2 Constructor & Destructor Documentation

- #### 7.39.2.1 Xapian::QueryParserError::QueryParserError (const std::string &msg\_, const std::string &context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.



### 7.39.2.2 Xapian::QueryParserError::QueryParserError (const std::string & msg\_, int *errno*\_) [inline]

Construct from message and errno value.

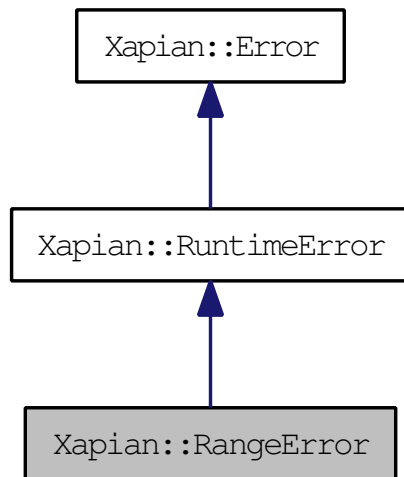
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.40 Xapian::RangeError Class Reference

[RangeError](#) indicates an attempt to access outside the bounds of a container.

Inheritance diagram for Xapian::RangeError:



### Public Member Functions

- [RangeError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_ = 0)

*General purpose constructor which allows setting errno.*

- [RangeError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.40.1 Detailed Description

[RangeError](#) indicates an attempt to access outside the bounds of a container.

### 7.40.2 Constructor & Destructor Documentation

#### 7.40.2.1 Xapian::RangeError::RangeError (const std::string & msg\_, const std::string & context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.

### 7.40.2.2 Xapian::RangeError::RangeError (const std::string & *msg\_*, int *errno\_*) [inline]

Construct from message and errno value.

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

- [xapian/error.h](#)

## 7.41 Xapian::RSet Class Reference

A relevance set (R-Set).

### Public Member Functions

- [RSet](#) (const [RSet](#) &rset)  
*Copy constructor.*
- void [operator=](#) (const [RSet](#) &rset)  
*Assignment operator.*
- [RSet](#) ()  
*Default constructor.*
- [~RSet](#) ()  
*Destructor.*
- [Xapian::doccount size](#) () const  
*The number of documents in this R-Set.*
- bool [empty](#) () const  
*Test if this R-Set is empty.*
- void [add\\_document](#) ([Xapian::docid](#) did)  
*Add a document to the relevance set.*
- void [add\\_document](#) (const [Xapian::MSetIterator](#) &i)  
*Add a document to the relevance set.*
- void [remove\\_document](#) ([Xapian::docid](#) did)  
*Remove a document from the relevance set.*
- void [remove\\_document](#) (const [Xapian::MSetIterator](#) &i)  
*Remove a document from the relevance set.*
- bool [contains](#) ([Xapian::docid](#) did) const  
*Test if a given document in the relevance set.*
- bool [contains](#) (const [Xapian::MSetIterator](#) &i) const  
*Test if a given document in the relevance set.*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

## Public Attributes

- Xapian::Internal::RefCntPtr< Internal > **internal**

### 7.41.1 Detailed Description

A relevance set (R-Set).

This is the set of documents which are marked as relevant, for use in modifying the term weights, and in performing query expansion.

### 7.41.2 Constructor & Destructor Documentation

#### 7.41.2.1 Xapian::RSet::RSet (const RSet & *rset*)

Copy constructor.

#### 7.41.2.2 Xapian::RSet::RSet ()

Default constructor.

#### 7.41.2.3 Xapian::RSet::~~RSet ()

Destructor.

### 7.41.3 Member Function Documentation

#### 7.41.3.1 void Xapian::RSet::add\_document (const Xapian::MSetIterator & *i*) [inline]

Add a document to the relevance set.

References add\_document().

Referenced by add\_document().

#### 7.41.3.2 void Xapian::RSet::add\_document (Xapian::docid *did*)

Add a document to the relevance set.

#### 7.41.3.3 bool Xapian::RSet::contains (const Xapian::MSetIterator & *i*) const [inline]

Test if a given document in the relevance set.

References contains().

Referenced by contains().

#### **7.41.3.4    bool Xapian::RSet::contains (Xapian::docid *did*) const**

Test if a given document in the relevance set.

#### **7.41.3.5    bool Xapian::RSet::empty () const**

Test if this R-Set is empty.

#### **7.41.3.6    std::string Xapian::RSet::get\_description () const**

Return a string describing this object.

#### **7.41.3.7    void Xapian::RSet::operator= (const RSet & *rset*)**

Assignment operator.

#### **7.41.3.8    void Xapian::RSet::remove\_document (const Xapian::MSetIterator & *i*) [inline]**

Remove a document from the relevance set.

References remove\_document().

Referenced by remove\_document().

#### **7.41.3.9    void Xapian::RSet::remove\_document (Xapian::docid *did*)**

Remove a document from the relevance set.

#### **7.41.3.10   Xapian::doccount Xapian::RSet::size () const**

The number of documents in this R-Set.

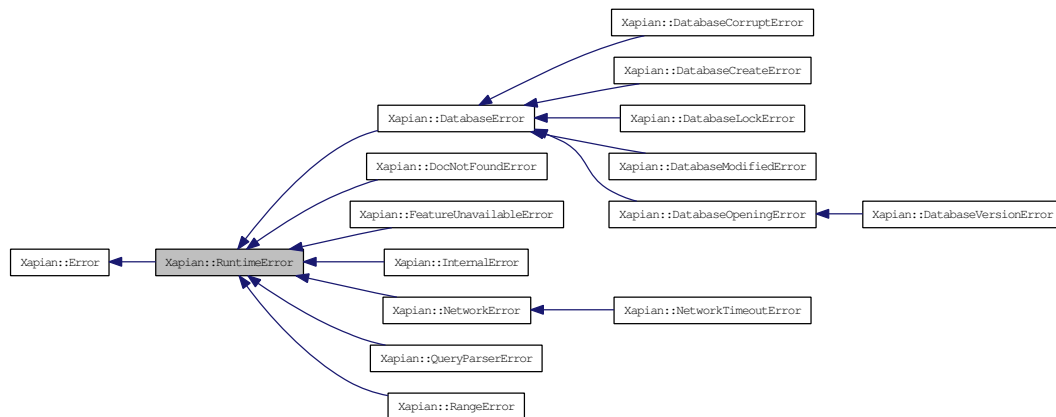
The documentation for this class was generated from the following file:

- xapian/[enquire.h](#)

## 7.42 Xapian::RuntimeError Class Reference

The base class for exceptions indicating errors only detectable at runtime.

Inheritance diagram for Xapian::RuntimeError:



### 7.42.1 Detailed Description

The base class for exceptions indicating errors only detectable at runtime.

A subclass of [RuntimeError](#) will be thrown if [Xapian](#) detects an error which is exception derived from [RuntimeError](#) is thrown when an error is caused by problems with the data or environment rather than a programming mistake.

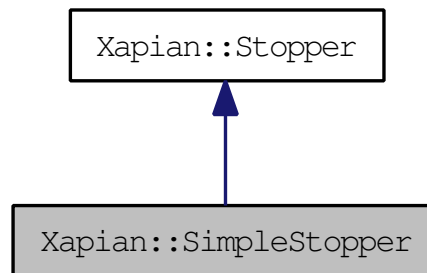
The documentation for this class was generated from the following file:

- [xapian/error.h](#)

## 7.43 Xapian::SimpleStopper Class Reference

Simple implementation of [Stopper](#) class - this will suit most users.

Inheritance diagram for Xapian::SimpleStopper:



### Public Member Functions

- [SimpleStopper](#) ()  
*Default constructor.*
- `template<class Iterator >`  
[SimpleStopper](#) (Iterator begin, Iterator end)  
*Initialise from a pair of iterators.*
- `void` [add](#) (const std::string &word)  
*Add a single stop word.*
- `virtual bool` [operator\(\)](#) (const std::string &term) const  
*Is term a stop-word?*
- `virtual` [~SimpleStopper](#) ()  
*Destructor.*
- `virtual std::string` [get\\_description](#) () const  
*Return a string describing this object.*

### 7.43.1 Detailed Description

Simple implementation of [Stopper](#) class - this will suit most users.



## 7.43.2 Constructor & Destructor Documentation

### 7.43.2.1 Xapian::SimpleStopper::SimpleStopper () [inline]

Default constructor.

### 7.43.2.2 template<class Iterator > Xapian::SimpleStopper::SimpleStopper (Iterator *begin*, Iterator *end*) [inline]

Initialise from a pair of iterators.

### 7.43.2.3 virtual Xapian::SimpleStopper::~SimpleStopper () [inline, virtual]

Destructor.

## 7.43.3 Member Function Documentation

### 7.43.3.1 void Xapian::SimpleStopper::add (const std::string & *word*) [inline]

Add a single stop word.

### 7.43.3.2 virtual std::string Xapian::SimpleStopper::get\_description () const [virtual]

Return a string describing this object.

Reimplemented from [Xapian::Stopper](#).

### 7.43.3.3 virtual bool Xapian::SimpleStopper::operator() (const std::string & *term*) const [inline, virtual]

Is term a stop-word?

Implements [Xapian::Stopper](#).

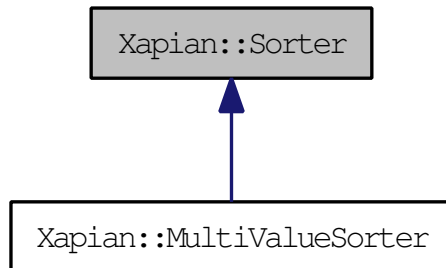
The documentation for this class was generated from the following file:

- [xapian/queryparser.h](#)

## 7.44 Xapian::Sorter Class Reference

Virtual base class for sorter functor.

Inheritance diagram for Xapian::Sorter:



### Public Member Functions

- virtual std::string [operator\(\)](#) (const [Xapian::Document](#) &doc) const =0  
*This method takes a [Document](#) object and builds a sort key from it.*
- virtual [~Sorter](#) ()  
*Virtual destructor, because we have virtual methods.*

### 7.44.1 Detailed Description

Virtual base class for sorter functor.

### 7.44.2 Constructor & Destructor Documentation

#### 7.44.2.1 virtual Xapian::Sorter::~~Sorter () [virtual]

Virtual destructor, because we have virtual methods.

### 7.44.3 Member Function Documentation

#### 7.44.3.1 virtual std::string Xapian::Sorter::operator() (const Xapian::Document & doc) const [pure virtual]

This method takes a [Document](#) object and builds a sort key from it.

Documents are then ordered by a string compare on the sort keys.

Implemented in [Xapian::MultiValueSorter](#).

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

- [xapian/sorter.h](#)

## 7.45 Xapian::Stem Class Reference

Class representing a stemming algorithm.

### Public Member Functions

- [Stem](#) (const [Stem](#) &o)  
*Copy constructor.*
- void [operator=](#) (const [Stem](#) &o)  
*Assignment.*
- [Stem](#) ()  
*Construct a [Xapian::Stem](#) object which doesn't change terms.*
- [Stem](#) (const std::string &language)  
*Construct a [Xapian::Stem](#) object for a particular language.*
- [~Stem](#) ()  
*Destructor.*
- std::string [operator\(\)](#) (const std::string &word) const  
*[Stem](#) a word.*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

### Static Public Member Functions

- static std::string [get\\_available\\_languages](#) ()  
*Return a list of available languages.*

#### 7.45.1 Detailed Description

Class representing a stemming algorithm.

#### 7.45.2 Constructor & Destructor Documentation

##### 7.45.2.1 Xapian::Stem::Stem (const Stem & o)

Copy constructor.

### 7.45.2.2 Xapian::Stem::Stem ()

Construct a [Xapian::Stem](#) object which doesn't change terms.

Equivalent to [Stem](#)("none").

### 7.45.2.3 Xapian::Stem::Stem (const std::string & language) [explicit]

Construct a [Xapian::Stem](#) object for a particular language.

#### Parameters:

*language* Either the English name for the language or the two letter ISO639 code.

The following language names are understood (aliases follow the name):

- none - don't stem terms
- danish (da)
- dutch (nl)
- english (en) - Martin Porter's 2002 revision of his stemmer
- english\_lovins (lovins) - Lovin's stemmer
- english\_porter (porter) - Porter's stemmer as described in his 1980 paper
- finnish (fi)
- french (fr)
- german (de)
- german2 - Normalises umlauts and ß
- hungarian (hu)
- italian (it)
- kraaij\_pohlmann - A different Dutch stemmer
- norwegian (nb, nn, no)
- portuguese (pt)
- romanian (ro)
- russian (ru)
- spanish (es)
- swedish (sv)
- turkish (tr)

#### Exceptions:

[Xapian::InvalidArgumentError](#) is thrown if language isn't recognised.

#### 7.45.2.4 Xapian::Stem::~~Stem ()

Destructor.

### 7.45.3 Member Function Documentation

#### 7.45.3.1 static std::string Xapian::Stem::get\_available\_languages () [static]

Return a list of available languages.

Each stemmer is only included once in the list (not once for each alias). The name included is the English name of the language.

The list is returned as a string, with language names separated by spaces. This is a static method, so a [Xapian::Stem](#) object is not required for this operation.

#### 7.45.3.2 std::string Xapian::Stem::get\_description () const

Return a string describing this object.

#### 7.45.3.3 std::string Xapian::Stem::operator() (const std::string & word) const

[Stem](#) a word.

##### Parameters:

*word* a word to stem.

##### Returns:

the stem

#### 7.45.3.4 void Xapian::Stem::operator= (const Stem & o)

Assignment.

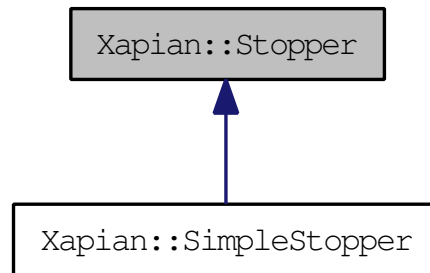
The documentation for this class was generated from the following file:

- [xapian/stem.h](#)

## 7.46 Xapian::Stopper Class Reference

Base class for stop-word decision functor.

Inheritance diagram for Xapian::Stopper:



### Public Member Functions

- virtual bool [operator\(\)](#) (const std::string &term) const =0  
*Is term a stop-word?*
- virtual [~Stopper](#) ()  
*Class has virtual methods, so provide a virtual destructor.*
- virtual std::string [get\\_description](#) () const  
*Return a string describing this object.*

### 7.46.1 Detailed Description

Base class for stop-word decision functor.

### 7.46.2 Constructor & Destructor Documentation

#### 7.46.2.1 virtual Xapian::Stopper::~~Stopper () [inline, virtual]

Class has virtual methods, so provide a virtual destructor.

### 7.46.3 Member Function Documentation

#### 7.46.3.1 virtual std::string Xapian::Stopper::get\_description () const [virtual]

Return a string describing this object.

Reimplemented in [Xapian::SimpleStopper](#).

#### 7.46.3.2 **virtual bool Xapian::Stopper::operator() (const std::string & *term*)** **const** [pure virtual]

Is term a stop-word?

Implemented in [Xapian::SimpleStopper](#).

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

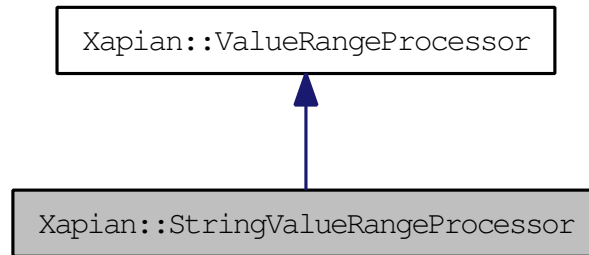
- [xapian/queryparser.h](#)



## 7.47 Xapian::StringValueRangeProcessor Class Reference

Handle a string range.

Inheritance diagram for Xapian::StringValueRangeProcessor:



### Public Member Functions

- [StringValueRangeProcessor](#) ([Xapian::valueno](#) valno\_)  
*Constructor.*
- [Xapian::valueno operator\(\)](#) (std::string &, std::string &)  
*Any strings are valid as begin and end.*

#### 7.47.1 Detailed Description

Handle a string range.

The end points can be any strings.

#### 7.47.2 Constructor & Destructor Documentation

##### 7.47.2.1 Xapian::StringValueRangeProcessor::StringValueRangeProcessor (Xapian::valueno valno\_) [inline]

Constructor.

##### Parameters:

*valno\_* The value number to return from operator().

### 7.47.3 Member Function Documentation

#### 7.47.3.1 `Xapian::value` `Xapian::StringValueRangeProcessor::operator()` (`std::string &`, `std::string &`) [`inline`, `virtual`]

Any strings are valid as begin and end.

Implements [Xapian::ValueRangeProcessor](#).

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

- [xapian/queryparser.h](#)

## 7.48 Xapian::TermGenerator Class Reference

Parses a piece of text and generate terms.

### Public Types

- enum [flags](#) { [FLAG\\_SPELLING](#) = 128 }  
*Flags to OR together and pass to [TermGenerator::set\\_flags\(\)](#).*

### Public Member Functions

- [TermGenerator](#) (const [TermGenerator](#) &o)  
*Copy constructor.*
- [TermGenerator](#) & operator= (const [TermGenerator](#) &o)  
*Assignment.*
- [TermGenerator](#) ()  
*Default constructor.*
- [~TermGenerator](#) ()  
*Destructor.*
- void [set\\_stemmer](#) (const [Xapian::Stem](#) &stemmer)  
*Set the [Xapian::Stem](#) object to be used for generating stemmed terms.*
- void [set\\_stopper](#) (const [Xapian::Stopper](#) \*stop=NULL)  
*Set the [Xapian::Stopper](#) object to be used for identifying stopwords.*
- void [set\\_document](#) (const [Xapian::Document](#) &doc)  
*Set the current document.*
- const [Xapian::Document](#) & [get\\_document](#) () const  
*Get the current document.*
- void [set\\_database](#) (const [Xapian::WritableDatabase](#) &db)  
*Set the database to index spelling data to.*
- [flags](#) [set\\_flags](#) ([flags](#) toggle, [flags](#) mask=[flags](#)(0))  
*Set flags.*
- void [index\\_text](#) (const [Xapian::Utf8Iterator](#) &itor, [Xapian::termcount](#) weight=1, const std::string &prefix="")  
*Index some text.*

- void [index\\_text](#) (const std::string &text, [Xapian::termcount](#) weight=1, const std::string &prefix="")  
*Index some text in a std::string.*
- void [index\\_text\\_without\\_positions](#) (const [Xapian::Utf8Iterator](#) &itor, [Xapian::termcount](#) weight=1, const std::string &prefix="")  
*Index some text without positional information.*
- void [index\\_text\\_without\\_positions](#) (const std::string &text, [Xapian::termcount](#) weight=1, const std::string &prefix="")  
*Index some text in a std::string without positional information.*
- void [increase\\_termpos](#) ([Xapian::termcount](#) delta=100)  
*Increase the termpos used by index\_text by delta.*
- [Xapian::termcount](#) [get\\_termpos](#) () const  
*Get the current term position.*
- void [set\\_termpos](#) ([Xapian::termcount](#) termpos)  
*Set the current term position.*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

### 7.48.1 Detailed Description

Parses a piece of text and generate terms.

This module takes a piece of text and parses it to produce words which are then used to generate suitable terms for indexing. The terms generated are suitable for use with [Query](#) objects produced by the [QueryParser](#) class.

### 7.48.2 Member Enumeration Documentation

#### 7.48.2.1 enum [Xapian::TermGenerator::flags](#)

Flags to OR together and pass to [TermGenerator::set\\_flags\(\)](#).

**Enumerator:**

***FLAG\_SPELLING*** Index data required for spelling correction.

### 7.48.3 Constructor & Destructor Documentation

#### 7.48.3.1 Xapian::TermGenerator::TermGenerator (const TermGenerator & o)

Copy constructor.

#### 7.48.3.2 Xapian::TermGenerator::TermGenerator ()

Default constructor.

#### 7.48.3.3 Xapian::TermGenerator::~~TermGenerator ()

Destructor.

### 7.48.4 Member Function Documentation

#### 7.48.4.1 std::string Xapian::TermGenerator::get\_description () const

Return a string describing this object.

#### 7.48.4.2 const Xapian::Document& Xapian::TermGenerator::get\_document () const

Get the current document.

#### 7.48.4.3 Xapian::termcount Xapian::TermGenerator::get\_termpos () const

Get the current term position.

#### 7.48.4.4 void Xapian::TermGenerator::increase\_termpos (Xapian::termcount *delta* = 100)

Increase the termpos used by index\_text by *delta*.

This can be used to prevent phrase searches from spanning two unconnected blocks of text (e.g. the title and body text).

#### 7.48.4.5 void Xapian::TermGenerator::index\_text (const std::string & *text*, Xapian::termcount *weight* = 1, const std::string & *prefix* = "") [inline]

Index some text in a std::string.

#### Parameters:

*weight* The wdf increment (default 1).

*prefix* The term prefix to use (default is no prefix).

**7.48.4.6** `void Xapian::TermGenerator::index_text (const Xapian::Utf8Iterator & itor, Xapian::termcount weight = 1, const std::string & prefix = "")`

Index some text.

**Parameters:**

*weight* The wdf increment (default 1).

*prefix* The term prefix to use (default is no prefix).

**7.48.4.7** `void Xapian::TermGenerator::index_text_without_positions (const std::string & text, Xapian::termcount weight = 1, const std::string & prefix = "") [inline]`

Index some text in a std::string without positional information.

Just like `index_text`, but no positional information is generated. This means that the database will be significantly smaller, but that phrase searching and NEAR won't be supported.

**7.48.4.8** `void Xapian::TermGenerator::index_text_without_positions (const Xapian::Utf8Iterator & itor, Xapian::termcount weight = 1, const std::string & prefix = "")`

Index some text without positional information.

Just like `index_text`, but no positional information is generated. This means that the database will be significantly smaller, but that phrase searching and NEAR won't be supported.

**7.48.4.9** `TermGenerator& Xapian::TermGenerator::operator= (const TermGenerator & o)`

Assignment.

**7.48.4.10** `void Xapian::TermGenerator::set_database (const Xapian::WritableDatabase & db)`

Set the database to index spelling data to.

**7.48.4.11** `void Xapian::TermGenerator::set_document (const Xapian::Document & doc)`

Set the current document.

**7.48.4.12** `flags Xapian::TermGenerator::set_flags (flags toggle, flags mask = flags (0))`

Set flags.

The new value of flags is: (flags & mask) ^ toggle

To just set the flags, pass the new flags in toggle and the default value for mask.

**Parameters:**

*toggle* Flags to XOR.

*mask* Flags to AND with first.

**Returns:**

The old flags setting.

**7.48.4.13** `void Xapian::TermGenerator::set_stemmer (const Xapian::Stem & stemmer)`

Set the [Xapian::Stem](#) object to be used for generating stemmed terms.

**7.48.4.14** `void Xapian::TermGenerator::set_stopper (const Xapian::Stopper * stop = NULL)`

Set the [Xapian::Stopper](#) object to be used for identifying stopwords.

Stemmed forms of stopwords aren't indexed, but unstemmed forms still are so that searches for phrases including stop words still work.

**7.48.4.15** `void Xapian::TermGenerator::set_termpos (Xapian::termcount termpos)`

Set the current term position.

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

- [xapian/termgenerator.h](#)

## 7.49 Xapian::TermIterator Class Reference

An iterator pointing to items in a list of terms.

### Public Types

- typedef std::input\_iterator\_tag [iterator\\_category](#)  
*Allow use as an STL iterator.*
- typedef std::string [value\\_type](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::termcount\\_diff](#) [difference\\_type](#)  
*Allow use as an STL iterator.*
- typedef std::string \* [pointer](#)  
*Allow use as an STL iterator.*
- typedef std::string & [reference](#)  
*Allow use as an STL iterator.*

### Public Member Functions

- **TermIterator** (Internal \*internal\_)
- [TermIterator](#) ()  
*Default constructor - for declaring an uninitialised iterator.*
- [~TermIterator](#) ()  
*Destructor.*
- [TermIterator](#) (const [TermIterator](#) &other)  
*Copying is allowed.*
- void [operator=](#) (const [TermIterator](#) &other)  
*Assignment is allowed.*
- std::string [operator\\*](#) () const  
*Return the current term.*
- [TermIterator](#) & [operator++](#) ()
- TermNameWrapper [operator++](#) (int)
- void [skip\\_to](#) (const std::string &name)  
*Advance the iterator to the specified term.*
- [Xapian::termcount](#) [get\\_wdf](#) () const



*Return the wdf of the current term (if meaningful).*

- [Xapian::doccount get\\_termfreq \(\)](#) const  
*Return the term frequency of the current term (if meaningful).*
- [Xapian::termcount positionlist\\_count \(\)](#) const  
*Return length of positionlist for current term.*
- [PositionIterator positionlist\\_begin \(\)](#) const  
*Return [PositionIterator](#) pointing to start of positionlist for current term.*
- [PositionIterator positionlist\\_end \(\)](#) const  
*Return [PositionIterator](#) pointing to end of positionlist for current term.*
- [std::string get\\_description \(\)](#) const  
*Return a string describing this object.*

## Public Attributes

- [Xapian::Internal::RefCountPtr< Internal > internal](#)

### 7.49.1 Detailed Description

An iterator pointing to items in a list of terms.

### 7.49.2 Member Typedef Documentation

#### 7.49.2.1 `typedef Xapian::termcount_diff Xapian::TermIterator::difference_type`

Allow use as an STL iterator.

#### 7.49.2.2 `typedef std::input_iterator_tag Xapian::TermIterator::iterator_category`

Allow use as an STL iterator.

#### 7.49.2.3 `typedef std::string* Xapian::TermIterator::pointer`

Allow use as an STL iterator.

#### 7.49.2.4 `typedef std::string& Xapian::TermIterator::reference`

Allow use as an STL iterator.

#### 7.49.2.5 `typedef std::string Xapian::TermIterator::value_type`

Allow use as an STL iterator.

### 7.49.3 Constructor & Destructor Documentation

#### 7.49.3.1 `Xapian::TermIterator::TermIterator ()`

Default constructor - for declaring an uninitialised iterator.

#### 7.49.3.2 `Xapian::TermIterator::~~TermIterator ()`

Destructor.

#### 7.49.3.3 `Xapian::TermIterator::TermIterator (const TermIterator & other)`

Copying is allowed.

The internals are reference counted, so copying is also cheap.

### 7.49.4 Member Function Documentation

#### 7.49.4.1 `std::string Xapian::TermIterator::get_description () const`

Return a string describing this object.

#### 7.49.4.2 `Xapian::doccount Xapian::TermIterator::get_termfreq () const`

Return the term frequency of the current term (if meaningful).

The term frequency is the number of documents which a term indexes.

#### 7.49.4.3 `Xapian::termcount Xapian::TermIterator::get_wdf () const`

Return the wdf of the current term (if meaningful).

The wdf (within document frequency) is the number of occurrences of a term in a particular document.

#### 7.49.4.4 `std::string Xapian::TermIterator::operator* () const`

Return the current term.

**7.49.4.5 void Xapian::TermIterator::operator= (const TermIterator & *other*)**

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

**7.49.4.6 PositionIterator Xapian::TermIterator::positionlist\_begin () const**

Return [PositionIterator](#) pointing to start of positionlist for current term.

**7.49.4.7 Xapian::termcount Xapian::TermIterator::positionlist\_count () const**

Return length of positionlist for current term.

**7.49.4.8 PositionIterator Xapian::TermIterator::positionlist\_end () const**  
[inline]

Return [PositionIterator](#) pointing to end of positionlist for current term.

**7.49.4.9 void Xapian::TermIterator::skip\_to (const std::string & *tname*)**

Advance the iterator to the specified term.

If the specified term isn't in the list, position ourselves on the first term after it (or at\_end() if no greater terms are present).

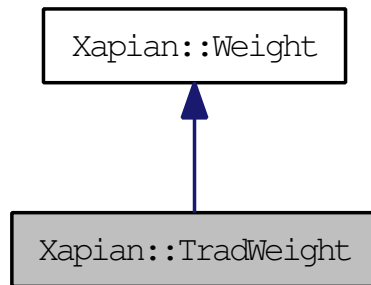
The documentation for this class was generated from the following file:

- [xapian/termiterator.h](#)

## 7.50 Xapian::TradWeight Class Reference

Traditional probabilistic weighting scheme.

Inheritance diagram for Xapian::TradWeight:



### Public Member Functions

- [TradWeight](#) (double k)  
*Construct a [TradWeight](#).*
- [TradWeight](#) \* [clone](#) () const  
*Return a new weight object of this type.*
- std::string [name](#) () const  
*Name of the weighting scheme.*
- std::string [serialise](#) () const  
*Serialise object parameters into a string.*
- [TradWeight](#) \* [unserialise](#) (const std::string &s) const  
*Create object given string serialisation returned by [serialise\(\)](#).*
- [Xapian::weight](#) [get\\_sumpart](#) ([Xapian::termcount](#) wdf, [Xapian::doclength](#) len) const  
*Get a weight which is part of the sum over terms being performed.*
- [Xapian::weight](#) [get\\_maxpart](#) () const  
*Gets the maximum value that [get\\_sumpart\(\)](#) may return.*
- [Xapian::weight](#) [get\\_sumextra](#) ([Xapian::doclength](#) len) const  
*Get an extra weight for a document to add to the sum calculated over the query terms.*
- [Xapian::weight](#) [get\\_maxextra](#) () const  
*Gets the maximum value that [get\\_sumextra\(\)](#) may return.*

- bool [get\\_sumpart\\_needs\\_doclength](#) () const  
*return false if the weight object doesn't need doclength*

### 7.50.1 Detailed Description

Traditional probabilistic weighting scheme.

This class implements the Traditional Probabilistic Weighting scheme, as described by the early papers on Probabilistic Retrieval. BM25 generally gives better results.

The Traditional weighting scheme formula is

$$\sum_t \frac{f_{t,d}}{k \cdot L_d + f_{t,d}} \cdot w_t$$

where

- $w_t$  is the termweight of term t
- $f_{t,d}$  is the within document frequency of term t in document d
- $L_d$  is the normalised length of document d
- $k$  is a user specifiable parameter

TradWeight(k) is equivalent to BM25Weight(k, 0, 0, 1, 0), except that the latter returns weights (k+1) times larger.

### 7.50.2 Constructor & Destructor Documentation

#### 7.50.2.1 Xapian::TradWeight::TradWeight (double k) [inline, explicit]

Construct a [TradWeight](#).

#### Parameters:

- $k$  parameter governing the importance of within document frequency and document length - any non-negative number (0 meaning to ignore wdf and doc length when calculating weights). Default is 1.

### 7.50.3 Member Function Documentation

#### 7.50.3.1 TradWeight\* Xapian::TradWeight::clone () const [virtual]

Return a new weight object of this type.

A subclass called `FooWeight` taking parameters `param1` and `param2` should implement this as:

```
virtual FooWeight * clone() const { return new FooWeight(param1, param2); }
```

Implements [Xapian::Weight](#).

#### 7.50.3.2 `Xapian::weight Xapian::TradWeight::get_maxextra () const` [virtual]

Gets the maximum value that [get\\_sumextra\(\)](#) may return.

This is used in optimising searches.

Implements [Xapian::Weight](#).

#### 7.50.3.3 `Xapian::weight Xapian::TradWeight::get_maxpart () const` [virtual]

Gets the maximum value that [get\\_sumpart\(\)](#) may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implements [Xapian::Weight](#).

#### 7.50.3.4 `Xapian::weight Xapian::TradWeight::get_sumextra (Xapian::doclength len) const` [virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

##### Parameters:

*len* the (unnormalised) document length.

Implements [Xapian::Weight](#).

#### 7.50.3.5 `Xapian::weight Xapian::TradWeight::get_sumpart (Xapian::termcount wdf, Xapian::doclength len) const` [virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

##### Parameters:

*wdf* the within document frequency of the term.

*len* the (unnormalised) document length.

Implements [Xapian::Weight](#).

#### 7.50.3.6 `bool Xapian::TradWeight::get_sumpart_needs_doclength () const` [virtual]

return false if the weight object doesn't need doclength

Reimplemented from [Xapian::Weight](#).

#### 7.50.3.7 `std::string Xapian::TradWeight::name () const` [virtual]

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implements [Xapian::Weight](#).

#### 7.50.3.8 `std::string Xapian::TradWeight::serialise () const` [virtual]

Serialise object parameters into a string.

Implements [Xapian::Weight](#).

#### 7.50.3.9 `TradWeight* Xapian::TradWeight::unserialise (const std::string & s) const` [virtual]

Create object given string serialisation returned by [serialise\(\)](#).

Implements [Xapian::Weight](#).

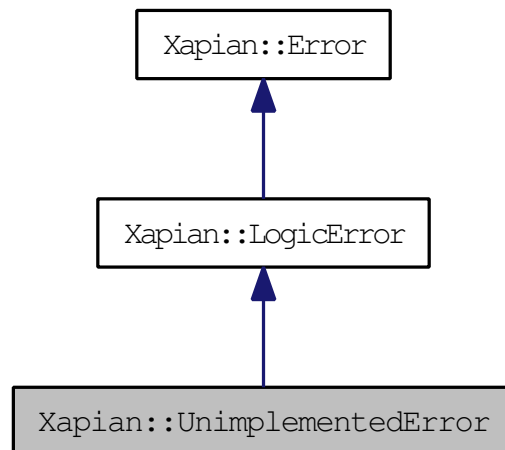
The documentation for this class was generated from the following file:

- [xapian/enquire.h](#)

## 7.51 Xapian::UnimplementedError Class Reference

[UnimplementedError](#) indicates an attempt to use an unimplemented feature.

Inheritance diagram for Xapian::UnimplementedError:



### Public Member Functions

- [UnimplementedError](#) (const std::string &msg\_, const std::string &context\_="", int errno\_=0)

*General purpose constructor which allows setting errno.*

- [UnimplementedError](#) (const std::string &msg\_, int errno\_)

*Construct from message and errno value.*

### 7.51.1 Detailed Description

[UnimplementedError](#) indicates an attempt to use an unimplemented feature.

### 7.51.2 Constructor & Destructor Documentation

- 7.51.2.1** [Xapian::UnimplementedError::UnimplementedError](#) (const std::string & msg\_, const std::string & context\_ = "", int errno\_ = 0) [inline, explicit]

General purpose constructor which allows setting errno.



### 7.51.2.2 Xapian::UnimplementedError::UnimplementedError (const std::string & *msg\_*, int *errno\_*) [inline]

Construct from message and errno value.

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

- [xapian/error.h](#)

## 7.52 Xapian::Utf8Iterator Class Reference

An iterator which returns [Unicode](#) character values from a UTF-8 encoded string.

### Public Types

- typedef std::input\_iterator\_tag [iterator\\_category](#)  
*We implement the semantics of an STL input\_iterator.*
- typedef unsigned [value\\_type](#)  
*We implement the semantics of an STL input\_iterator.*
- typedef size\_t [difference\\_type](#)  
*We implement the semantics of an STL input\_iterator.*
- typedef const unsigned \* [pointer](#)  
*We implement the semantics of an STL input\_iterator.*
- typedef const unsigned & [reference](#)  
*We implement the semantics of an STL input\_iterator.*

### Public Member Functions

- const char \* [raw](#) () const  
*Return the raw const char \* pointer for the current position.*
- size\_t [left](#) () const  
*Return the number of bytes left in the iterator's buffer.*
- void [assign](#) (const char \*p\_, size\_t len)  
*Assign a new string to the iterator.*
- void [assign](#) (const std::string &s)  
*Assign a new string to the iterator.*
- [Utf8Iterator](#) (const char \*p\_)  
*Create an iterator given a pointer to a null terminated string.*
- [Utf8Iterator](#) (const char \*p\_, size\_t len)  
*Create an iterator given a pointer and a length.*
- [Utf8Iterator](#) (const std::string &s)  
*Create an iterator given a string.*
- [Utf8Iterator](#) ()

*Create an iterator which is at the end of its iteration.*

- unsigned `operator*` () const  
*Get the current [Unicode](#) character value pointed to by the iterator.*
- `Utf8Iterator operator++` (int)  
*Move forward to the next [Unicode](#) character.*
- `Utf8Iterator & operator++` ()  
*Move forward to the next [Unicode](#) character.*
- bool `operator==` (const `Utf8Iterator` &other) const  
*Test two `Utf8Iterators` for equality.*
- bool `operator!=` (const `Utf8Iterator` &other) const  
*Test two `Utf8Iterators` for inequality.*

### 7.52.1 Detailed Description

An iterator which returns [Unicode](#) character values from a UTF-8 encoded string.

### 7.52.2 Member Typedef Documentation

#### 7.52.2.1 `typedef size_t Xapian::Utf8Iterator::difference_type`

We implement the semantics of an STL `input_iterator`.

#### 7.52.2.2 `typedef std::input_iterator_tag Xapian::Utf8Iterator::iterator_category`

We implement the semantics of an STL `input_iterator`.

#### 7.52.2.3 `typedef const unsigned* Xapian::Utf8Iterator::pointer`

We implement the semantics of an STL `input_iterator`.

#### 7.52.2.4 `typedef const unsigned& Xapian::Utf8Iterator::reference`

We implement the semantics of an STL `input_iterator`.

#### 7.52.2.5 `typedef unsigned Xapian::Utf8Iterator::value_type`

We implement the semantics of an STL `input_iterator`.

## 7.52.3 Constructor & Destructor Documentation

### 7.52.3.1 `Xapian::Utf8Iterator::Utf8Iterator (const char * p_)` `[explicit]`

Create an iterator given a pointer to a null terminated string.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

#### Parameters:

*p\_* A pointer to the start of the null terminated string to read.

### 7.52.3.2 `Xapian::Utf8Iterator::Utf8Iterator (const char * p_, size_t len)` `[inline]`

Create an iterator given a pointer and a length.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

#### Parameters:

*p\_* A pointer to the start of the string to read.

*len* The length of the string to read.

### 7.52.3.3 `Xapian::Utf8Iterator::Utf8Iterator (const std::string & s)` `[inline]`

Create an iterator given a string.

The iterator will return characters from the start of the string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

#### Parameters:

*s* The string to read. Must not be modified while the iteration is in progress.

### 7.52.3.4 `Xapian::Utf8Iterator::Utf8Iterator ()` `[inline]`

Create an iterator which is at the end of its iteration.

This can be compared to another iterator to check if the other iterator has reached its end.

## 7.52.4 Member Function Documentation

### 7.52.4.1 void Xapian::Utf8Iterator::assign (const std::string & *s*) [inline]

Assign a new string to the iterator.

The iterator will forget the string it was iterating through, and return characters from the start of the new string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

#### Parameters:

*s* The string to read. Must not be modified while the iteration is in progress.

References assign().

Referenced by assign().

### 7.52.4.2 void Xapian::Utf8Iterator::assign (const char \* *p\_*, size\_t *len*) [inline]

Assign a new string to the iterator.

The iterator will forget the string it was iterating through, and return characters from the start of the new string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

#### Parameters:

*p\_* A pointer to the start of the string to read.

*len* The length of the string to read.

### 7.52.4.3 size\_t Xapian::Utf8Iterator::left () const [inline]

Return the number of bytes left in the iterator's buffer.

### 7.52.4.4 bool Xapian::Utf8Iterator::operator!= (const Utf8Iterator & *other*) const [inline]

Test two Utf8Iterators for inequality.

#### Returns:

true iff the iterators do not point to the same position.

### 7.52.4.5 unsigned Xapian::Utf8Iterator::operator\* () const

Get the current [Unicode](#) character value pointed to by the iterator.

Returns unsigned(-1) if the iterator has reached the end of its buffer.

**7.52.4.6** `Utf8Iterator& Xapian::Utf8Iterator::operator++ ()` `[inline]`

Move forward to the next [Unicode](#) character.

**Returns:**

A reference to this object.

**7.52.4.7** `Utf8Iterator Xapian::Utf8Iterator::operator++ (int)` `[inline]`

Move forward to the next [Unicode](#) character.

**Returns:**

An iterator pointing to the position before the move.

**7.52.4.8** `bool Xapian::Utf8Iterator::operator== (const Utf8Iterator & other)`  
`const` `[inline]`

Test two Utf8Iterators for equality.

**Returns:**

true iff the iterators point to the same position.

**7.52.4.9** `const char* Xapian::Utf8Iterator::raw () const` `[inline]`

Return the raw const char \* pointer for the current position.

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

- [xapian/unicode.h](#)

## 7.53 Xapian::ValueIterator Class Reference

An iterator pointing to values associated with a document.

### Public Types

- typedef std::input\_iterator\_tag [iterator\\_category](#)  
*Allow use as an STL iterator.*
- typedef std::string [value\\_type](#)  
*Allow use as an STL iterator.*
- typedef [Xapian::valueno\\_diff](#) [difference\\_type](#)  
*Allow use as an STL iterator.*
- typedef std::string \* [pointer](#)  
*Allow use as an STL iterator.*
- typedef std::string & [reference](#)  
*Allow use as an STL iterator.*

### Public Member Functions

- [ValueIterator](#) ()  
*Create an uninitialised iterator; this cannot be used, but is convenient syntactically.*
- [ValueIterator](#) (const [ValueIterator](#) &other)  
*Copying is allowed (and is cheap).*
- void [operator=](#) (const [ValueIterator](#) &other)  
*Assignment is allowed (and is cheap).*
- [ValueIterator](#) & [operator++](#) ()  
*Advance the iterator.*
- [ValueIterator](#) [operator++](#) (int)  
*Advance the iterator (postfix variant).*
- const std::string & [operator\\*](#) () const  
*Get the value for the current position.*
- const std::string \* [operator →](#) () const  
*Get the value for the current position.*
- [Xapian::valueno](#) [get\\_valueno](#) () const

*Get the number of the value at the current position.*

- `std::string get\_description () const`  
*Return a string describing this object.*

## Friends

- `bool operator== (const ValueIterator &a, const ValueIterator &b)`  
*Equality test for [ValueIterator](#) objects.*
- `bool operator!= (const ValueIterator &a, const ValueIterator &b)`  
*Inequality test for [ValueIterator](#) objects.*

## 7.53.1 Detailed Description

An iterator pointing to values associated with a document.

## 7.53.2 Member Typedef Documentation

### 7.53.2.1 `typedef Xapian::valueno_diff Xapian::ValueIterator::difference_type`

Allow use as an STL iterator.

### 7.53.2.2 `typedef std::input_iterator_tag Xapian::ValueIterator::iterator_category`

Allow use as an STL iterator.

### 7.53.2.3 `typedef std::string* Xapian::ValueIterator::pointer`

Allow use as an STL iterator.

### 7.53.2.4 `typedef std::string& Xapian::ValueIterator::reference`

Allow use as an STL iterator.

### 7.53.2.5 `typedef std::string Xapian::ValueIterator::value_type`

Allow use as an STL iterator.



### 7.53.3 Constructor & Destructor Documentation

#### 7.53.3.1 Xapian::ValueIterator::ValueIterator () [inline]

Create an uninitialised iterator; this cannot be used, but is convenient syntactically.

#### 7.53.3.2 Xapian::ValueIterator::ValueIterator (const ValueIterator & *other*) [inline]

Copying is allowed (and is cheap).

### 7.53.4 Member Function Documentation

#### 7.53.4.1 std::string Xapian::ValueIterator::get\_description () const

Return a string describing this object.

#### 7.53.4.2 Xapian::valueno Xapian::ValueIterator::get\_valueno () const

Get the number of the value at the current position.

#### 7.53.4.3 const std::string& Xapian::ValueIterator::operator\* () const

Get the value for the current position.

#### 7.53.4.4 ValueIterator Xapian::ValueIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

#### 7.53.4.5 ValueIterator& Xapian::ValueIterator::operator++ () [inline]

Advance the iterator.

#### 7.53.4.6 const std::string\* Xapian::ValueIterator::operator → () const

Get the value for the current position.

#### 7.53.4.7 void Xapian::ValueIterator::operator= (const ValueIterator & *other*) [inline]

Assignment is allowed (and is cheap).

## 7.53.5 Friends And Related Function Documentation

### 7.53.5.1 `bool operator!= (const ValueIterator & a, const ValueIterator & b)` [friend]

Inequality test for [ValueIterator](#) objects.

### 7.53.5.2 `bool operator== (const ValueIterator & a, const ValueIterator & b)` [friend]

Equality test for [ValueIterator](#) objects.

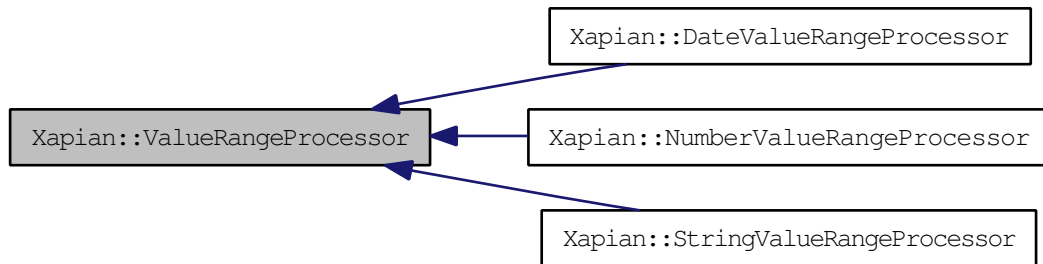
The documentation for this class was generated from the following file:

- xapian/[valueiterator.h](#)

## 7.54 Xapian::ValueRangeProcessor Struct Reference

Base class for value range processors.

Inheritance diagram for Xapian::ValueRangeProcessor:



### Public Member Functions

- virtual [~ValueRangeProcessor](#) ()  
*Destructor.*
- virtual [Xapian::valueno operator\(\)](#) (std::string &begin, std::string &end)=0  
*Check for a valid range of this type.*

#### 7.54.1 Detailed Description

Base class for value range processors.

#### 7.54.2 Constructor & Destructor Documentation

- 7.54.2.1** virtual [Xapian::ValueRangeProcessor::~~ValueRangeProcessor](#) ()  
[virtual]

Destructor.

#### 7.54.3 Member Function Documentation

- 7.54.3.1** virtual [Xapian::valueno Xapian::ValueRangeProcessor::operator\(\)](#)  
(std::string &begin, std::string &end) [pure virtual]

Check for a valid range of this type.

If this [ValueRangeProcessor](#) recognises BEGIN..END it returns the value number to range filter on. Otherwise it returns [Xapian::BAD\\_VALUENO](#).

Implemented in [Xapian::StringValueRangeProcessor](#),  
[Xapian::DateValueRangeProcessor](#), and [Xapian::NumberValueRangeProcessor](#).

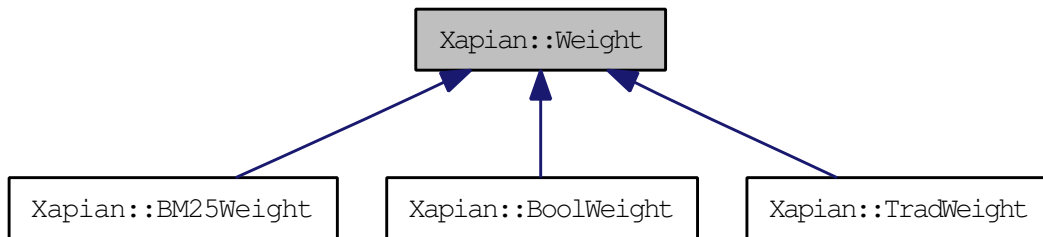
The documentation for this struct was generated from the following file:

- [xapian/queryparser.h](#)

## 7.55 Xapian::Weight Class Reference

Abstract base class for weighting schemes.

Inheritance diagram for Xapian::Weight:



### Public Member Functions

- `Weight * create` (const Internal \*internal\_, `Xapian::doclength` querysize\_, `Xapian::termcount` wqf\_, const std::string &name\_) const  
Create a new weight object of the same type as this and initialise it with the specified statistics.
- virtual std::string `name` () const =0  
Name of the weighting scheme.
- virtual std::string `serialise` () const =0  
Serialise object parameters into a string.
- virtual `Weight * unserialise` (const std::string &s) const =0  
Create object given string serialisation returned by `serialise()`.
- virtual `Xapian::weight get_sumpart` (`Xapian::termcount` wdf, `Xapian::doclength` len) const =0  
Get a weight which is part of the sum over terms being performed.
- virtual `Xapian::weight get_maxpart` () const =0  
Gets the maximum value that `get_sumpart()` may return.
- virtual `Xapian::weight get_sumextra` (`Xapian::doclength` len) const =0  
Get an extra weight for a document to add to the sum calculated over the query terms.
- virtual `Xapian::weight get_maxextra` () const =0  
Gets the maximum value that `get_sumextra()` may return.
- virtual bool `get_sumpart_needs_doclength` () const  
return false if the weight object doesn't need doclength

## Protected Member Functions

- **Weight** (const [Weight](#) &)

## Protected Attributes

- const Internal \* **internal**
- [Xapian::doclength](#) **querysize**
- [Xapian::termcount](#) **wqf**
- std::string **tname**

### 7.55.1 Detailed Description

Abstract base class for weighting schemes.

### 7.55.2 Member Function Documentation

#### 7.55.2.1 **Weight\*** [Xapian::Weight::create](#) (const Internal \* *internal\_*, [Xapian::doclength](#) *querysize\_*, [Xapian::termcount](#) *wqf\_*, const std::string & *tname\_*) const

Create a new weight object of the same type as this and initialise it with the specified statistics.

You shouldn't call this method yourself - it's called by [Enquire](#).

#### Parameters:

*internal\_* Object to ask for collection statistics.

*querysize\_* [Query](#) size.

*wqf\_* Within query frequency of term this object is associated with.

*tname\_* Term which this object is associated with.

#### 7.55.2.2 **virtual** [Xapian::weight](#) [Xapian::Weight::get\\_maxextra](#) () const [pure virtual]

Gets the maximum value that [get\\_sumextra\(\)](#) may return.

This is used in optimising searches.

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

**7.55.2.3** `virtual Xapian::weight Xapian::Weight::get_maxpart () const` [pure virtual]

Gets the maximum value that `get_sumpart()` may return.

This is used in optimising searches, by having the postlist tree decay appropriately when parts of it can have limited, or no, further effect.

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

**7.55.2.4** `virtual Xapian::weight Xapian::Weight::get_sumextra (Xapian::doclength len) const` [pure virtual]

Get an extra weight for a document to add to the sum calculated over the query terms.

This returns a weight for a given document, and is used by some weighting schemes to account for influence such as document length.

**Parameters:**

*len* the (unnormalised) document length.

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

**7.55.2.5** `virtual Xapian::weight Xapian::Weight::get_sumpart (Xapian::termcount wdf, Xapian::doclength len) const` [pure virtual]

Get a weight which is part of the sum over terms being performed.

This returns a weight for a given term and document. These weights are summed to give a total weight for the document.

**Parameters:**

*wdf* the within document frequency of the term.

*len* the (unnormalised) document length.

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

**7.55.2.6** `virtual bool Xapian::Weight::get_sumpart_needs_doclength () const` [virtual]

return false if the weight object doesn't need doclength

Reimplemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

**7.55.2.7 virtual std::string Xapian::Weight::name () const** [pure virtual]

Name of the weighting scheme.

If the subclass is called FooWeight, this should return "Foo".

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

**7.55.2.8 virtual std::string Xapian::Weight::serialise () const** [pure virtual]

Serialise object parameters into a string.

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

**7.55.2.9 virtual Weight\* Xapian::Weight::unserialise (const std::string & s) const** [pure virtual]

Create object given string serialisation returned by [serialise\(\)](#).

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

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

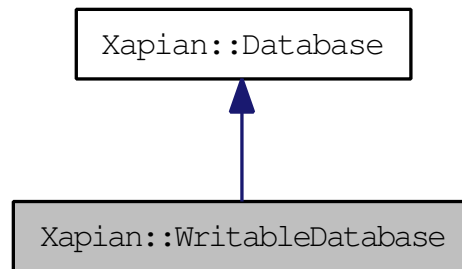
- [xapian/enquire.h](#)



## 7.56 Xapian::WritableDatabase Class Reference

This class provides read/write access to a database.

Inheritance diagram for Xapian::WritableDatabase:



### Public Member Functions

- virtual [~WritableDatabase](#) ()  
*Destroy this handle on the database.*
- [WritableDatabase](#) ()  
*Create an empty [WritableDatabase](#).*
- [WritableDatabase](#) (const std::string &path, int action)  
*Open a database for update, automatically determining the database backend to use.*
- [WritableDatabase](#) (const [WritableDatabase](#) &other)  
*Copying is allowed.*
- void [operator=](#) (const [WritableDatabase](#) &other)  
*Assignment is allowed.*
- void [flush](#) ()  
*Commit any pending modifications made to the database.*
- void [begin\\_transaction](#) (bool flushed=true)  
*Begin a transaction.*
- void [commit\\_transaction](#) ()  
*Complete the transaction currently in progress.*
- void [cancel\\_transaction](#) ()  
*Abort the transaction currently in progress, discarding the pending modifications made to the database.*

- [Xapian::docid add\\_document](#) (const [Xapian::Document](#) &document)  
*Add a new document to the database.*
- void [delete\\_document](#) ([Xapian::docid](#) did)  
*Delete a document from the database.*
- void [delete\\_document](#) (const std::string &unique\_term)  
*Delete any documents indexed by a term from the database.*
- void [replace\\_document](#) ([Xapian::docid](#) did, const [Xapian::Document](#) &document)  
*Replace a given document in the database.*
- [Xapian::docid replace\\_document](#) (const std::string &unique\_term, const [Xapian::Document](#) &document)  
*Replace any documents matching a term.*
- void [add\\_spelling](#) (const std::string &word, [Xapian::termcount](#) freqinc=1) const  
*Add a word to the spelling dictionary.*
- void [remove\\_spelling](#) (const std::string &word, [Xapian::termcount](#) freqdec=1) const  
*Remove a word from the spelling dictionary.*
- void [add\\_synonym](#) (const std::string &term, const std::string &synonym) const  
*Add a synonym for a term.*
- void [remove\\_synonym](#) (const std::string &term, const std::string &synonym) const  
*Remove a synonym for a term.*
- void [clear\\_synonyms](#) (const std::string &term) const  
*Remove all synonyms for a term.*
- void [set\\_metadata](#) (const std::string &key, const std::string &value)  
*Set the user-specified metadata associated with a given key.*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

### 7.56.1 Detailed Description

This class provides read/write access to a database.

## 7.56.2 Constructor & Destructor Documentation

### 7.56.2.1 virtual Xapian::WritableDatabase::~~WritableDatabase () [virtual]

Destroy this handle on the database.

If there are no copies of this object remaining, the database will be closed. If there are any transactions in progress these will be aborted as if `cancel_transaction` had been called.

### 7.56.2.2 Xapian::WritableDatabase::WritableDatabase ()

Create an empty [WritableDatabase](#).

### 7.56.2.3 Xapian::WritableDatabase::WritableDatabase (const std::string & *path*, int *action*)

Open a database for update, automatically determining the database backend to use.

If the database is to be created, [Xapian](#) will try to create the directory indicated by *path* if it doesn't already exist (but only the leaf directory, not recursively).

#### Parameters:

*path* directory that the database is stored in.

*action* one of:

- [Xapian::DB\\_CREATE\\_OR\\_OPEN](#) open for read/write; create if no db exists
- [Xapian::DB\\_CREATE](#) create new database; fail if db exists
- [Xapian::DB\\_CREATE\\_OR\\_OVERWRITE](#) overwrite existing db; create if none exists
- [Xapian::DB\\_OPEN](#) open for read/write; fail if no db exists

#### Exceptions:

[Xapian::DatabaseCorruptError](#) will be thrown if the database is in a corrupt state.

[Xapian::DatabaseLockError](#) will be thrown if a lock couldn't be acquired on the database.

### 7.56.2.4 Xapian::WritableDatabase::WritableDatabase (const WritableDatabase & *other*)

Copying is allowed.

The internals are reference counted, so copying is cheap.

### 7.56.3 Member Function Documentation

#### 7.56.3.1 `Xapian::docid Xapian::WritableDatabase::add_document (const Xapian::Document & document)`

Add a new document to the database.

This method adds the specified document to the database, returning a newly allocated document ID. Automatically allocated document IDs come from a per-database monotonically increasing counter, so IDs from deleted documents won't be reused.

If you want to specify the document ID to be used, you should call [replace\\_document\(\)](#) instead.

Note that changes to the database won't be immediately committed to disk; see [flush\(\)](#) for more details.

As with all database modification operations, the effect is atomic: the document will either be fully added, or the document fails to be added and an exception is thrown (possibly at a later time when [flush\(\)](#) is called or the database is closed).

##### Parameters:

*document* The new document to be added.

##### Returns:

The document ID of the newly added document.

##### Exceptions:

[Xapian::DatabaseError](#) will be thrown if a problem occurs while writing to the database.

[Xapian::DatabaseCorruptError](#) will be thrown if the database is in a corrupt state.

#### 7.56.3.2 `void Xapian::WritableDatabase::add_spelling (const std::string & word, Xapian::termcount freqinc = 1) const`

Add a word to the spelling dictionary.

If the word is already present, its frequency is increased.

##### Parameters:

*word* The word to add.

*freqinc* How much to increase its frequency by (default 1).

#### 7.56.3.3 `void Xapian::WritableDatabase::add_synonym (const std::string & term, const std::string & synonym) const`

Add a synonym for a term.

If *synonym* is already a synonym for *term*, then no action is taken.

#### 7.56.3.4 void Xapian::WritableDatabase::begin\_transaction (bool *flushed* = true)

Begin a transaction.

In [Xapian](#) a transaction is a group of modifications to the database which are linked such that either all will be applied simultaneously or none will be applied at all. Even in the case of a power failure, this characteristic should be preserved (as long as the filesystem isn't corrupted, etc).

A transaction is started with [begin\\_transaction\(\)](#) and can either be committed by calling [commit\\_transaction\(\)](#) or aborted by calling [cancel\\_transaction\(\)](#).

By default, a transaction implicitly calls [flush\(\)](#) before and after so that the modifications stand and fall without affecting modifications before or after.

The downside of these implicit calls to [flush\(\)](#) is that small transactions can harm indexing performance in the same way that explicitly calling [flush\(\)](#) frequently can.

If you're applying atomic groups of changes and only wish to ensure that each group is either applied or not applied, then you can prevent the automatic flush before and after the transaction by starting the transaction with [begin\\_transaction\(false\)](#). However, if [cancel\\_transaction](#) is called (or if [commit\\_transaction](#) isn't called before the [WritableDatabase](#) object is destroyed) then any changes which were pending before the transaction began will also be discarded.

Transactions aren't currently supported by the [InMemory](#) backend.

#### Exceptions:

[Xapian::UnimplementedError](#) will be thrown if transactions are not available for this database type.

[Xapian::InvalidOperationError](#) will be thrown if this is called at an invalid time, such as when a transaction is already in progress.

#### 7.56.3.5 void Xapian::WritableDatabase::cancel\_transaction ()

Abort the transaction currently in progress, discarding the pending modifications made to the database.

If an error occurs in this method, an exception will be thrown, but the transaction will be cancelled anyway.

#### Exceptions:

[Xapian::DatabaseError](#) will be thrown if a problem occurs while modifying the database.

[Xapian::DatabaseCorruptError](#) will be thrown if the database is in a corrupt state.

*Xapian::InvalidOperationError* will be thrown if a transaction is not currently in progress.

*Xapian::UnimplementedError* will be thrown if transactions are not available for this database type.

#### 7.56.3.6 void Xapian::WritableDatabase::clear\_synonyms (const std::string & term) const

Remove all synonyms for a term.

If *term* has no synonyms, no action is taken.

#### 7.56.3.7 void Xapian::WritableDatabase::commit\_transaction ()

Complete the transaction currently in progress.

If this method completes successfully and this is a flushed transaction, all the database modifications made during the transaction will have been committed to the database.

If an error occurs, an exception will be thrown, and none of the modifications made to the database during the transaction will have been applied to the database.

In all cases the transaction will no longer be in progress.

#### Exceptions:

*Xapian::DatabaseError* will be thrown if a problem occurs while modifying the database.

*Xapian::DatabaseCorruptError* will be thrown if the database is in a corrupt state.

*Xapian::InvalidOperationError* will be thrown if a transaction is not currently in progress.

*Xapian::UnimplementedError* will be thrown if transactions are not available for this database type.

#### 7.56.3.8 void Xapian::WritableDatabase::delete\_document (const std::string & unique\_term)

Delete any documents indexed by a term from the database.

This method removes any documents indexed by the specified term from the database.

A major use is for convenience when UIDs from another system are mapped to terms in *Xapian*, although this method has other uses (for example, you could add a "deletion date" term to documents at index time and use this method to delete all documents due for deletion on a particular date).

#### Parameters:

*unique\_term* The term to remove references to.

**Exceptions:**

*Xapian::DatabaseError* will be thrown if a problem occurs while writing to the database.

*Xapian::DatabaseCorruptError* will be thrown if the database is in a corrupt state.

**7.56.3.9 void Xapian::WritableDatabase::delete\_document (Xapian::docid *did*)**

Delete a document from the database.

This method removes the document with the specified document ID from the database.

Note that changes to the database won't be immediately committed to disk; see [flush\(\)](#) for more details.

As with all database modification operations, the effect is atomic: the document will either be fully removed, or the document fails to be removed and an exception is thrown (possibly at a later time when [flush\(\)](#) is called or the database is closed).

**Parameters:**

*did* The document ID of the document to be removed.

**Exceptions:**

*Xapian::DatabaseError* will be thrown if a problem occurs while writing to the database.

*Xapian::DatabaseCorruptError* will be thrown if the database is in a corrupt state.

**7.56.3.10 void Xapian::WritableDatabase::flush ()**

Commit any pending modifications made to the database.

For efficiency reasons, when performing multiple updates to a database it is best (indeed, almost essential) to make as many modifications as memory will permit in a single pass through the database. To ensure this, [Xapian](#) batches up modifications.

Flush may be called at any time to ensure that the modifications which have been made are written to disk: if the flush succeeds, all the preceding modifications will have been written to disk.

If any of the modifications fail, an exception will be thrown and the database will be left in a state in which each separate addition, replacement or deletion operation has either been fully performed or not performed at all: it is then up to the application to work out which operations need to be repeated.

It's not valid to call [flush\(\)](#) within a transaction.

Beware of calling [flush\(\)](#) too frequently: this will make indexing take much longer.

Note that [flush\(\)](#) need not be called explicitly: it will be called automatically when the database is closed, or when a sufficient number of modifications have been made. By default, this is every 10000 documents added, deleted, or modified. This value is rather conservative, and if you have a machine with plenty of memory, you can improve indexing throughput dramatically by setting `XAPIAN_FLUSH_THRESHOLD` in the environment to a larger value.

#### Exceptions:

[\*Xapian::DatabaseError\*](#) will be thrown if a problem occurs while modifying the database.

[\*Xapian::DatabaseCorruptError\*](#) will be thrown if the database is in a corrupt state.

#### 7.56.3.11 `std::string Xapian::WritableDatabase::get_description () const` [virtual]

Return a string describing this object.

Reimplemented from [Xapian::Database](#).

#### 7.56.3.12 `void Xapian::WritableDatabase::operator= (const WritableDatabase & other)`

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

Note that only an [WritableDatabase](#) may be assigned to an [WritableDatabase](#): an attempt to assign a [Database](#) is caught at compile-time.

#### 7.56.3.13 `void Xapian::WritableDatabase::remove_spelling (const std::string & word, Xapian::termcount freqdec = 1) const`

Remove a word from the spelling dictionary.

The word's frequency is decreased, and if would become zero or less then the word is removed completely.

#### Parameters:

*word* The word to remove.

*freqdec* How much to decrease its frequency by (default 1).

#### 7.56.3.14 `void Xapian::WritableDatabase::remove_synonym (const std::string & term, const std::string & synonym) const`

Remove a synonym for a term.

If *synonym* isn't a synonym for *term*, then no action is taken.



### 7.56.3.15 Xapian::docid Xapian::WritableDatabase::replace\_document (const std::string & *unique\_term*, const Xapian::Document & *document*)

Replace any documents matching a term.

This method replaces any documents indexed by the specified term with the specified document. If any documents are indexed by the term, the lowest document ID will be used for the document, otherwise a new document ID will be generated as for `add_document`.

One common use is to allow UUIDs from another system to easily be mapped to terms in Xapian. Note that this method doesn't automatically add `unique_term` as a term, so you'll need to call `document.add_term(unique_term)` first when using `replace_document()` in this way.

Another possible use is to allow groups of documents to be marked for later deletion - for example, you could add a "deletion date" term to documents at index time and use this method to easily and efficiently delete all documents due for deletion on a particular date.

Note that changes to the database won't be immediately committed to disk; see `flush()` for more details.

As with all database modification operations, the effect is atomic: the document(s) will either be fully replaced, or the document(s) fail to be replaced and an exception is thrown (possibly at a later time when `flush()` is called or the database is closed).

#### Parameters:

*unique\_term* The "unique" term.

*document* The new document.

#### Returns:

The document ID that document was given.

#### Exceptions:

[\*Xapian::DatabaseError\*](#) will be thrown if a problem occurs while writing to the database.

[\*Xapian::DatabaseCorruptError\*](#) will be thrown if the database is in a corrupt state.

### 7.56.3.16 void Xapian::WritableDatabase::replace\_document (Xapian::docid *did*, const Xapian::Document & *document*)

Replace a given document in the database.

This method replaces the document with the specified document ID. If document ID *did* isn't currently used, the document will be added with document ID *did*.

The monotonic counter used for automatically allocating document IDs is increased so that the next automatically allocated document ID will be *did* + 1. Be aware that

if you use this method to specify a high document ID for a new document, and also use `WritableDatabase::add_document()`, `Xapian` may get to a state where this counter wraps around and will be unable to automatically allocate document IDs!

Note that changes to the database won't be immediately committed to disk; see `flush()` for more details.

As with all database modification operations, the effect is atomic: the document will either be fully replaced, or the document fails to be replaced and an exception is thrown (possibly at a later time when `flush()` is called or the database is closed).

#### Parameters:

*did* The document ID of the document to be replaced.

*document* The new document.

#### Exceptions:

`Xapian::DatabaseError` will be thrown if a problem occurs while writing to the database.

`Xapian::DatabaseCorruptError` will be thrown if the database is in a corrupt state.

#### 7.56.3.17 void Xapian::WritableDatabase::set\_metadata (const std::string &key, const std::string &value)

Set the user-specified metadata associated with a given key.

This method sets the metadata value associated with a given key. If there is already a metadata value stored in the database with the same key, the old value is replaced. If you want to delete an existing item of metadata, just set its value to the empty string.

User-specified metadata allows you to store arbitrary information in the form of (key,tag) pairs.

There's no hard limit on the number of metadata items, or the size of the metadata values. Metadata keys have a limited length, which depends on the backend. We recommend limiting them to 200 bytes. Empty keys are not valid, and specifying one will cause an exception.

Metadata modifications are committed to disk in the same way as modifications to the documents in the database are: i.e., modifications are atomic, and won't be committed to disk immediately (see `flush()` for more details). This allows metadata to be used to link databases with versioned external resources by storing the appropriate version number in a metadata item.

You can also use the metadata to store arbitrary extra information associated with terms, documents, or postings by encoding the termname and/or document id into the metadata key.

#### Parameters:

*key* The key of the metadata item to set.

*value* The value of the metadata item to set.

**Exceptions:**

*Xapian::DatabaseError* will be thrown if a problem occurs while writing to the database.

*Xapian::DatabaseCorruptError* will be thrown if the database is in a corrupt state.

*Xapian::InvalidArgumentError* will be thrown if the key supplied is empty.

*Xapian::UnimplementedError* will be thrown if the database backend in use doesn't support user-specified metadata.

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

- [xapian/database.h](#)



## Chapter 8

# File Documentation

### 8.1 xapian/error.h File Reference

Hierarchy of classes which [Xapian](#) can throw as exceptions.

#### Classes

- class [Xapian::Error](#)  
*All exceptions thrown by [Xapian](#) are subclasses of [Xapian::Error](#).*
- class [Xapian::LogicError](#)  
*The base class for exceptions indicating errors in the program logic.*
- class [Xapian::RuntimeError](#)  
*The base class for exceptions indicating errors only detectable at runtime.*
- class [Xapian::AssertionError](#)  
*[AssertionError](#) is thrown if a logical assertion inside [Xapian](#) fails.*
- class [Xapian::InvalidArgumentError](#)  
*[InvalidArgumentError](#) indicates an invalid parameter value was passed to the API.*
- class [Xapian::InvalidOperationError](#)  
*[InvalidOperationError](#) indicates the API was used in an invalid way.*
- class [Xapian::UnimplementedError](#)  
*[UnimplementedError](#) indicates an attempt to use an unimplemented feature.*
- class [Xapian::DatabaseError](#)  
*[DatabaseError](#) indicates some sort of database related error.*

- class [Xapian::DatabaseCorruptError](#)  
*DatabaseCorruptError* indicates database corruption was detected.
- class [Xapian::DatabaseCreateError](#)  
*DatabaseCreateError* indicates a failure to create a database.
- class [Xapian::DatabaseLockError](#)  
*DatabaseLockError* indicates failure to lock a database.
- class [Xapian::DatabaseModifiedError](#)  
*DatabaseModifiedError* indicates a database was modified.
- class [Xapian::DatabaseOpeningError](#)  
*DatabaseOpeningError* indicates failure to open a database.
- class [Xapian::DatabaseVersionError](#)  
*DatabaseVersionError* indicates that a database is in an unsupported format.
- class [Xapian::DocNotFoundError](#)  
*Indicates an attempt to access a document not present in the database.*
- class [Xapian::FeatureUnavailableError](#)  
*Indicates an attempt to use a feature which is unavailable.*
- class [Xapian::InternalError](#)  
*InternalError* indicates a runtime problem of some sort.
- class [Xapian::NetworkError](#)  
*Indicates a problem communicating with a remote database.*
- class [Xapian::NetworkTimeoutError](#)  
*Indicates a timeout expired while communicating with a remote database.*
- class [Xapian::QueryParserError](#)  
*Indicates a query string can't be parsed.*
- class [Xapian::RangeError](#)  
*RangeError* indicates an attempt to access outside the bounds of a container.

## Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### **8.1.1 Detailed Description**

Hierarchy of classes which [Xapian](#) can throw as exceptions.

## 8.2 xapian/version.h File Reference

Define preprocessor symbols for the library version.

### Defines

- `#define XAPIAN_VERSION "1.0.23"`  
*The version of [Xapian](#) as a C string literal.*
- `#define XAPIAN_MAJOR_VERSION 1`  
*The major component of the [Xapian](#) version.*
- `#define XAPIAN_MINOR_VERSION 0`  
*The minor component of the [Xapian](#) version.*
- `#define XAPIAN_REVISION 23`  
*The revision component of the [Xapian](#) version.*
- `#define XAPIAN_HAS_FLINT_BACKEND 1`  
*XAPIAN\_HAS\_FLINT\_BACKEND Defined if the flint backend is enabled.*
- `#define XAPIAN_HAS_QUARTZ_BACKEND 1`  
*XAPIAN\_HAS\_QUARTZ\_BACKEND Defined if the quartz backend is enabled.*
- `#define XAPIAN_HAS_INMEMORY_BACKEND 1`  
*XAPIAN\_HAS\_INMEMORY\_BACKEND Defined if the inmemory backend is enabled.*
- `#define XAPIAN_HAS_REMOTE_BACKEND 1`  
*XAPIAN\_HAS\_REMOTE\_BACKEND Defined if the remote backend is enabled.*

### 8.2.1 Detailed Description

Define preprocessor symbols for the library version.

### 8.2.2 Define Documentation

#### 8.2.2.1 `#define XAPIAN_HAS_FLINT_BACKEND 1`

XAPIAN\_HAS\_FLINT\_BACKEND Defined if the flint backend is enabled.



**8.2.2.2 #define XAPIAN\_HAS\_INMEMORY\_BACKEND 1**

XAPIAN\_HAS\_INMEMORY\_BACKEND Defined if the inmemory backend is enabled.

**8.2.2.3 #define XAPIAN\_HAS\_QUARTZ\_BACKEND 1**

XAPIAN\_HAS\_QUARTZ\_BACKEND Defined if the quartz backend is enabled.

**8.2.2.4 #define XAPIAN\_HAS\_REMOTE\_BACKEND 1**

XAPIAN\_HAS\_REMOTE\_BACKEND Defined if the remote backend is enabled.

**8.2.2.5 #define XAPIAN\_MAJOR\_VERSION 1**

The major component of the [Xapian](#) version.

E.g. for [Xapian](#) 1.0.14 this would be: 1

**8.2.2.6 #define XAPIAN\_MINOR\_VERSION 0**

The minor component of the [Xapian](#) version.

E.g. for [Xapian](#) 1.0.14 this would be: 0

**8.2.2.7 #define XAPIAN\_REVISION 23**

The revision component of the [Xapian](#) version.

E.g. for [Xapian](#) 1.0.14 this would be: 14

**8.2.2.8 #define XAPIAN\_VERSION "1.0.23"**

The version of [Xapian](#) as a C string literal.

## 8.3 xapian.h File Reference

Public interfaces for the [Xapian](#) library.

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Functions

- const char \* [Xapian::version\\_string](#) ()  
*Report the version string of the library which the program is linked with.*
- const char \* [Xapian::xapian\\_version\\_string](#) ()  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*
- int [Xapian::major\\_version](#) ()  
*Report the major version of the library which the program is linked with.*
- int [Xapian::xapian\\_major\\_version](#) ()  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*
- int [Xapian::minor\\_version](#) ()  
*Report the minor version of the library which the program is linked with.*
- int [Xapian::xapian\\_minor\\_version](#) ()  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*
- int [Xapian::revision](#) ()  
*Report the revision of the library which the program is linked with.*
- int [Xapian::xapian\\_revision](#) ()  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*

#### 8.3.1 Detailed Description

Public interfaces for the [Xapian](#) library.

## 8.4 xapian/database.h File Reference

API for working with [Xapian](#) databases.

### Classes

- class [Xapian::Database](#)  
*This class is used to access a database, or a group of databases.*
- class [Xapian::WritableDatabase](#)  
*This class provides read/write access to a database.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Variables

- const int [Xapian::DB\\_CREATE\\_OR\\_OPEN](#) = 1  
*Open for read/write; create if no db exists.*
- const int [Xapian::DB\\_CREATE](#) = 2  
*Create a new database; fail if db exists.*
- const int [Xapian::DB\\_CREATE\\_OR\\_OVERWRITE](#) = 3  
*Overwrite existing db; create if none exists.*
- const int [Xapian::DB\\_OPEN](#) = 4  
*Open for read/write; fail if no db exists.*

#### 8.4.1 Detailed Description

API for working with [Xapian](#) databases.

## 8.5 xapian/dbfactory.h File Reference

Factory functions for constructing Database and WritableDatabase objects.

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*
- namespace [Xapian::Auto](#)  
*[Database](#) factory functions which determine the database type automatically.*
- namespace [Xapian::InMemory](#)  
*[Database](#) factory functions for the inmemory backend.*
- namespace [Xapian::Quartz](#)  
*[Database](#) factory functions for the quartz backend.*
- namespace [Xapian::Flint](#)  
*[Database](#) factory functions for the flint backend.*
- namespace [Xapian::Remote](#)  
*[Database](#) factory functions for the remote backend.*

### Functions

- Database [Xapian::Auto::open\\_stub](#) (const std::string &file)  
*Construct a [Database](#) object for a stub database file.*
- WritableDatabase [Xapian::InMemory::open](#) ()  
*Construct a [WritableDatabase](#) object for a new, empty [InMemory](#) database.*
- Database [Xapian::Quartz::open](#) (const std::string &dir)  
*Construct a [Database](#) object for read-only access to a [Quartz](#) database.*
- [Xapian::Quartz::WritableDatabaseopen](#) (const std::string &dir, int action, int block\_size=8192)  
*Construct a [Database](#) object for update access to a [Quartz](#) database.*
- Database [Xapian::Flint::open](#) (const std::string &dir)  
*Construct a [Database](#) object for read-only access to a [Flint](#) database.*
- WritableDatabase [Xapian::Flint::open](#) (const std::string &dir, int action, int block\_size=8192)

Construct a *Database* object for update access to a *Flint* database.

- Database `Xapian::Remote::open` (const std::string &host, unsigned int port, `Xapian::timeout` timeout=10000, `Xapian::timeout` connect\_timeout=10000)

Construct a *Database* object for read-only access to a remote database accessed via a TCP connection.

- WritableDatabase `Xapian::Remote::open_writable` (const std::string &host, unsigned int port, `Xapian::timeout` timeout=0, `Xapian::timeout` connect\_timeout=10000)

Construct a *WritableDatabase* object for update access to a remote database accessed via a TCP connection.

- Database `Xapian::Remote::open` (const std::string &program, const std::string &args, `Xapian::timeout` timeout=10000)

Construct a *Database* object for read-only access to a remote database accessed via a program.

- WritableDatabase `Xapian::Remote::open_writable` (const std::string &program, const std::string &args, `Xapian::timeout` timeout=0)

Construct a *WritableDatabase* object for update access to a remote database accessed via a program.

### 8.5.1 Detailed Description

Factory functions for constructing Database and WritableDatabase objects.

## 8.6 xapian/document.h File Reference

API for working with documents.

### Classes

- class [Xapian::Document](#)  
*A handle representing a document in a [Xapian](#) database.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### 8.6.1 Detailed Description

API for working with documents.

## 8.7 xapian/enquire.h File Reference

API for running queries.

### Classes

- class [Xapian::MSet](#)  
*A match set ([MSet](#)).*
- class [Xapian::MSetIterator](#)  
*An iterator pointing to items in an [MSet](#).*
- class [Xapian::ESet](#)  
*Class representing an ordered set of expand terms (an [ESet](#)).*
- class [Xapian::ESetIterator](#)  
*Iterate through terms in the [ESet](#).*
- class [Xapian::RSet](#)  
*A relevance set ([R-Set](#)).*
- class [Xapian::MatchDecider](#)  
*Base class for matcher decision functor.*
- class [Xapian::Enquire](#)  
*This class provides an interface to the information retrieval system for the purpose of searching.*
- class [Xapian::Weight](#)  
*Abstract base class for weighting schemes.*
- class [Xapian::BoolWeight](#)  
*Boolean weighting scheme (everything gets 0).*
- class [Xapian::BM25Weight](#)  
*BM25 weighting scheme.*
- class [Xapian::TradWeight](#)  
*Traditional probabilistic weighting scheme.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

## Functions

- bool [Xapian::operator==](#) (const MSetIterator &a, const MSetIterator &b)  
*Equality test for [MSetIterator](#) objects.*
- bool [Xapian::operator!=](#) (const MSetIterator &a, const MSetIterator &b)  
*Inequality test for [MSetIterator](#) objects.*
- bool [Xapian::operator==](#) (const ESetIterator &a, const ESetIterator &b)  
*Equality test for [ESetIterator](#) objects.*
- bool [Xapian::operator!=](#) (const ESetIterator &a, const ESetIterator &b)  
*Inequality test for [ESetIterator](#) objects.*

### 8.7.1 Detailed Description

API for running queries.



## 8.8 xapian/errorhandler.h File Reference

Decide if a [Xapian::Error](#) exception should be ignored.

### Classes

- class [Xapian::ErrorHandler](#)  
*Decide if a [Xapian::Error](#) exception should be ignored.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

#### 8.8.1 Detailed Description

Decide if a [Xapian::Error](#) exception should be ignored.

## 8.9 xapian/expanddecider.h File Reference

Allow rejection of terms during ESet generation.

### Classes

- class [Xapian::ExpandDecider](#)  
*Virtual base class for expand decider functor.*
- class [Xapian::ExpandDeciderAnd](#)  
*[ExpandDecider](#) subclass which rejects terms using two [ExpandDeciders](#).*
- class [Xapian::ExpandDeciderFilterTerms](#)  
*[ExpandDecider](#) subclass which rejects terms in a specified list.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### 8.9.1 Detailed Description

Allow rejection of terms during ESet generation.

## 8.10 xapian/positioniterator.h File Reference

Classes for iterating through position lists.

### Classes

- class [Xapian::PositionIterator](#)  
*An iterator pointing to items in a list of positions.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Functions

- bool [Xapian::operator==](#) (const PositionIterator &a, const PositionIterator &b)  
*Test equality of two PositionIterators.*
- bool [Xapian::operator!=](#) (const PositionIterator &a, const PositionIterator &b)  
*Test inequality of two PositionIterators.*

#### 8.10.1 Detailed Description

Classes for iterating through position lists.

## 8.11 xapian/postingiterator.h File Reference

Classes for iterating through posting lists.

### Classes

- class [Xapian::PostingIterator](#)  
*An iterator pointing to items in a list of postings.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Functions

- bool [Xapian::operator==](#) (const PostingIterator &a, const PostingIterator &b)  
*Test equality of two PostingIterators.*
- bool [Xapian::operator!=](#) (const PostingIterator &a, const PostingIterator &b)  
*Test inequality of two PostingIterators.*

#### 8.11.1 Detailed Description

Classes for iterating through posting lists.

## 8.12 xapian/query.h File Reference

Classes for representing a query.

### Classes

- class [Xapian::Query](#)  
*Class representing a query.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### 8.12.1 Detailed Description

Classes for representing a query.

## 8.13 xapian/queryparser.h File Reference

parsing a user query string to build a [Xapian::Query](#) object

### Classes

- class [Xapian::Stopper](#)  
*Base class for stop-word decision functor.*
- class [Xapian::SimpleStopper](#)  
*Simple implementation of [Stopper](#) class - this will suit most users.*
- struct [Xapian::ValueRangeProcessor](#)  
*Base class for value range processors.*
- class [Xapian::StringValueRangeProcessor](#)  
*Handle a string range.*
- class [Xapian::DateValueRangeProcessor](#)  
*Handle a date range.*
- class [Xapian::NumberValueRangeProcessor](#)  
*Handle a number range.*
- class [Xapian::QueryParser](#)  
*Build a [Xapian::Query](#) object from a user query string.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Functions

- `std::string Xapian::sortable\_serialise (double value)`  
*Convert a floating point number to a string, preserving sort order.*
- `double Xapian::sortable\_unserialise (const std::string &value)`  
*Convert a string encoded using [sortable\\_serialise](#) back to a floating point number.*

### **8.13.1 Detailed Description**

parsing a user query string to build a [Xapian::Query](#) object

## 8.14 xapian/sorter.h File Reference

Build sort keys for MSet ordering.

### Classes

- class [Xapian::Sorter](#)  
*Virtual base class for sorter functor.*
- class [Xapian::MultiValueSorter](#)  
*[Sorter](#) subclass which sorts by a several values.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

#### 8.14.1 Detailed Description

Build sort keys for MSet ordering.



## 8.15 xapian/stem.h File Reference

stemming algorithms

### Classes

- class [Xapian::Stem](#)  
*Class representing a stemming algorithm.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### 8.15.1 Detailed Description

stemming algorithms

## 8.16 xapian/termgenerator.h File Reference

parse free text and generate terms

### Classes

- class [Xapian::TermGenerator](#)  
*Parses a piece of text and generate terms.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

#### 8.16.1 Detailed Description

parse free text and generate terms

## 8.17 xapian/termiterator.h File Reference

Classes for iterating through term lists.

### Classes

- class [Xapian::TermIterator](#)  
*An iterator pointing to items in a list of terms.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Functions

- bool [Xapian::operator==](#) (const TermIterator &a, const TermIterator &b)  
*Equality test for [TermIterator](#) objects.*
- bool [Xapian::operator!=](#) (const TermIterator &a, const TermIterator &b)  
*Inequality test for [TermIterator](#) objects.*

#### 8.17.1 Detailed Description

Classes for iterating through term lists.

## 8.18 xapian/types.h File Reference

typedefs for [Xapian](#)

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Typedefs

- typedef unsigned [Xapian::doccount](#)  
*A count of documents.*
- typedef int [Xapian::doccount\\_diff](#)  
*A signed difference between two counts of documents.*
- typedef unsigned [Xapian::docid](#)  
*A unique identifier for a document.*
- typedef double [Xapian::doclength](#)  
*A normalised document length.*
- typedef int [Xapian::percent](#)  
*The percentage score for a document in an [MSet](#).*
- typedef unsigned [Xapian::termcount](#)  
*A counts of terms.*
- typedef int [Xapian::termcount\\_diff](#)  
*A signed difference between two counts of terms.*
- typedef unsigned [Xapian::termpos](#)  
*A term position within a document or query.*
- typedef int [Xapian::termpos\\_diff](#)  
*A signed difference between two term positions.*
- typedef unsigned [Xapian::timeout](#)  
*A timeout value in milliseconds.*
- typedef unsigned [Xapian::valueno](#)  
*The number for a value slot in a document.*

- typedef int [Xapian::valueno\\_diff](#)  
*A signed difference between two value slot numbers.*
- typedef double [Xapian::weight](#)  
*The weight of a document or term.*

## Variables

- const valueno [Xapian::BAD\\_VALUENO](#) = static\_cast<valueno>(-1)  
*Reserved value to indicate "no valueno".*

### 8.18.1 Detailed Description

typedefs for [Xapian](#)

## 8.19 xapian/unicode.h File Reference

Unicode and UTF-8 related classes and functions.

### Classes

- class [Xapian::Utf8Iterator](#)

*An iterator which returns [Unicode](#) character values from a UTF-8 encoded string.*

### Namespaces

- namespace [Xapian](#)

*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

- namespace [Xapian::Unicode](#)

*Functions associated with handling [Unicode](#) characters.*

### Enumerations

- enum [Xapian::Unicode::category](#) {  
UNASSIGNED,    UPPERCASE\_LETTER,    LOWERCASE\_LETTER,  
TITLECASE\_LETTER,  
MODIFIER\_LETTER,    OTHER\_LETTER,    NON\_SPACING\_MARK,  
ENCLOSING\_MARK,  
COMBINING\_SPACING\_MARK,    DECIMAL\_DIGIT\_NUMBER,  
LETTER\_NUMBER, OTHER\_NUMBER,  
SPACE\_SEPARATOR,    LINE\_SEPARATOR,    PARAGRAPH\_ -  
SEPARATOR, CONTROL,  
FORMAT,    PRIVATE\_USE,    SURROGATE,    CONNECTOR\_ -  
PUNCTUATION,  
DASH\_PUNCTUATION,    OPEN\_PUNCTUATION,    CLOSE\_ -  
PUNCTUATION, INITIAL\_QUOTE\_PUNCTUATION,  
FINAL\_QUOTE\_PUNCTUATION, OTHER\_PUNCTUATION, MATH\_ -  
SYMBOL, CURRENCY\_SYMBOL,  
MODIFIER\_SYMBOL, OTHER\_SYMBOL }

*Each Unicode character is in exactly one of these categories.*

## Functions

- int **Xapian::Unicode::Internal::get\_character\_info** (unsigned ch)
- int **Xapian::Unicode::Internal::get\_case\_type** (int info)
- category **Xapian::Unicode::Internal::get\_category** (int info)
- int **Xapian::Unicode::Internal::get\_delta** (int info)
- unsigned **Xapian::Unicode::nonascii\_to\_utf8** (unsigned ch, char \*buf)  
*Convert a single non-ASCII [Unicode](#) character to UTF-8.*
- unsigned **Xapian::Unicode::to\_utf8** (unsigned ch, char \*buf)  
*Convert a single [Unicode](#) character to UTF-8.*
- void **Xapian::Unicode::append\_utf8** (std::string &s, unsigned ch)  
*Append the UTF-8 representation of a single [Unicode](#) character to a std::string.*
- category **Xapian::Unicode::get\_category** (unsigned ch)  
*Return the category which a given [Unicode](#) character falls into.*
- bool **Xapian::Unicode::is\_wordchar** (unsigned ch)  
*Test if a given [Unicode](#) character is "word character".*
- bool **Xapian::Unicode::is\_whitespace** (unsigned ch)  
*Test if a given [Unicode](#) character is a whitespace character.*
- bool **Xapian::Unicode::is\_currency** (unsigned ch)  
*Test if a given [Unicode](#) character is a currency symbol.*
- unsigned **Xapian::Unicode::tolower** (unsigned ch)  
*Convert a [Unicode](#) character to lowercase.*
- unsigned **Xapian::Unicode::toupper** (unsigned ch)  
*Convert a [Unicode](#) character to uppercase.*
- std::string **Xapian::Unicode::tolower** (const std::string &term)  
*Convert a UTF-8 std::string to lowercase.*
- std::string **Xapian::Unicode::toupper** (const std::string &term)  
*Convert a UTF-8 std::string to uppercase.*

### 8.19.1 Detailed Description

Unicode and UTF-8 related classes and functions.

## 8.20 xapian/valueiterator.h File Reference

classes for iterating through values

### Classes

- class [Xapian::ValueIterator](#)  
*An iterator pointing to values associated with a document.*

### Namespaces

- namespace [Xapian](#)  
*The [Xapian](#) namespace contains public interfaces for the [Xapian](#) library.*

### Functions

- bool [Xapian::operator==](#) (const ValueIterator &a, const ValueIterator &b)  
*Equality test for [ValueIterator](#) objects.*
- bool [Xapian::operator!=](#) (const ValueIterator &a, const ValueIterator &b)  
*Inequality test for [ValueIterator](#) objects.*

#### 8.20.1 Detailed Description

classes for iterating through values



# Index

- ~Database
  - Xapian::Database, [51](#)
- ~Document
  - Xapian::Document, [77](#)
- ~ESet
  - Xapian::ESet, [101](#)
- ~Enquire
  - Xapian::Enquire, [85](#)
- ~ErrorHandler
  - Xapian::ErrorHandler, [98](#)
- ~ExpandDecider
  - Xapian::ExpandDecider, [107](#)
- ~MSet
  - Xapian::MSet, [125](#)
- ~MatchDecider
  - Xapian::MatchDecider, [121](#)
- ~PositionIterator
  - Xapian::PositionIterator, [144](#)
- ~PostingIterator
  - Xapian::PostingIterator, [148](#)
- ~Query
  - Xapian::Query, [153](#)
- ~QueryParser
  - Xapian::QueryParser, [159](#)
- ~RSet
  - Xapian::RSet, [169](#)
- ~SimpleStopper
  - Xapian::SimpleStopper, [173](#)
- ~Sorter
  - Xapian::Sorter, [174](#)
- ~Stem
  - Xapian::Stem, [177](#)
- ~Stopper
  - Xapian::Stopper, [179](#)
- ~TermGenerator
  - Xapian::TermGenerator, [185](#)
- ~TermIterator
  - Xapian::TermIterator, [190](#)
- ~ValueRangeProcessor
  - Xapian::ValueRangeProcessor, [207](#)
- ~WritableDatabase
  - Xapian::WritableDatabase, [215](#)
- add
  - Xapian::SimpleStopper, [173](#)
- add\_boolean\_prefix
  - Xapian::QueryParser, [159](#)
- add\_boolean\_term
  - Xapian::Document, [77](#)
- add\_database
  - Xapian::Database, [51](#)
- add\_document
  - Xapian::RSet, [169](#)
  - Xapian::WritableDatabase, [216](#)
- add\_posting
  - Xapian::Document, [77](#)
- add\_prefix
  - Xapian::QueryParser, [160](#)
- add\_spelling
  - Xapian::WritableDatabase, [216](#)
- add\_synonym
  - Xapian::WritableDatabase, [216](#)
- add\_term
  - Xapian::Document, [78](#)
- add\_value
  - Xapian::Document, [78](#)
- add\_valuerangeprocessor
  - Xapian::QueryParser, [160](#)
- allterms\_begin
  - Xapian::Database, [52](#)
- allterms\_end
  - Xapian::Database, [52](#)
- append\_utf8
  - Xapian::Unicode, [36](#)
- AssertionError
  - Xapian::AssertionError, [40](#)
- assign
  - Xapian::Utf8Iterator, [201](#)
- back
  - Xapian::ESet, [101](#)
  - Xapian::MSet, [125](#)

- BAD\_VALUENO
  - Xapian, [25](#)
- begin
  - Xapian::ESet, [101](#)
  - Xapian::MSet, [125](#)
- begin\_transaction
  - Xapian::WritableDatabase, [217](#)
- BM25Weight
  - Xapian::BM25Weight, [42](#)
- cancel\_transaction
  - Xapian::WritableDatabase, [217](#)
- category
  - Xapian::Unicode, [36](#)
- clear\_synonyms
  - Xapian::WritableDatabase, [218](#)
- clear\_terms
  - Xapian::Document, [78](#)
- clear\_values
  - Xapian::Document, [78](#)
- clone
  - Xapian::BM25Weight, [43](#)
  - Xapian::BoolWeight, [46](#)
  - Xapian::TradWeight, [193](#)
- commit\_transaction
  - Xapian::WritableDatabase, [218](#)
- const\_iterator
  - Xapian::MSet, [124](#)
- const\_reference
  - Xapian::MSet, [124](#)
- contains
  - Xapian::RSet, [169](#), [170](#)
- convert\_to\_percent
  - Xapian::MSet, [125](#), [126](#)
- create
  - Xapian::Weight, [210](#)
- Database
  - Xapian::Database, [51](#)
- DatabaseCorruptError
  - Xapian::DatabaseCorruptError, [59](#)
- DatabaseCreateError
  - Xapian::DatabaseCreateError, [61](#)
- DatabaseError
  - Xapian::DatabaseError, [62](#)
- DatabaseLockError
  - Xapian::DatabaseLockError, [64](#)
- DatabaseModifiedError
  - Xapian::DatabaseModifiedError, [66](#)
- DatabaseOpeningError
  - Xapian::DatabaseOpeningError, [68](#)
- DatabaseVersionError
  - Xapian::DatabaseVersionError, [70](#)
- DateValueRangeProcessor
  - Xapian::DateValueRangeProcessor, [71](#)
- DB\_CREATE
  - Xapian, [25](#)
- DB\_CREATE\_OR\_OPEN
  - Xapian, [25](#)
- DB\_CREATE\_OR\_OVERWRITE
  - Xapian, [25](#)
- DB\_OPEN
  - Xapian, [25](#)
- delete\_document
  - Xapian::WritableDatabase, [218](#), [219](#)
- difference\_type
  - Xapian::ESetIterator, [104](#)
  - Xapian::MSet, [124](#)
  - Xapian::MSetIterator, [131](#)
  - Xapian::PostingIterator, [147](#)
  - Xapian::TermIterator, [189](#)
  - Xapian::Utf8Iterator, [199](#)
  - Xapian::ValueIterator, [204](#)
- doccount
  - Xapian, [20](#)
- doccount\_diff
  - Xapian, [20](#)
- docid
  - Xapian, [20](#)
- doclength
  - Xapian, [20](#)
- DocNotFoundError
  - Xapian::DocNotFoundError, [73](#)
- Document
  - Xapian::Document, [77](#)
- empty
  - Xapian::ESet, [101](#)
  - Xapian::MSet, [126](#)
  - Xapian::Query, [154](#)
  - Xapian::RSet, [170](#)
- end
  - Xapian::ESet, [101](#)
  - Xapian::MSet, [126](#)
- Enquire
  - Xapian::Enquire, [85](#)
- ErrorHandler
  - Xapian::ErrorHandler, [98](#)
- ESet

- Xapian::ESet, [101](#)
- ESetIterator
  - Xapian::ESetIterator, [105](#)
- ExpandDeciderAnd
  - Xapian::ExpandDeciderAnd, [108](#)
- ExpandDeciderFilterTerms
  - Xapian::ExpandDeciderFilterTerms, [110](#)
- feature\_flag
  - Xapian::QueryParser, [157](#)
- FeatureUnavailableError
  - Xapian::FeatureUnavailableError, [112](#)
- fetch
  - Xapian::MSet, [126](#)
- FLAG\_AUTO\_MULTIWORD\_-SYNONYMS
  - Xapian::QueryParser, [158](#)
- FLAG\_AUTO\_SYNONYMS
  - Xapian::QueryParser, [158](#)
- FLAG\_BOOLEAN
  - Xapian::QueryParser, [158](#)
- FLAG\_BOOLEAN\_ANY\_CASE
  - Xapian::QueryParser, [158](#)
- FLAG\_DEFAULT
  - Xapian::QueryParser, [158](#)
- FLAG\_LOVEHATE
  - Xapian::QueryParser, [158](#)
- FLAG\_PARTIAL
  - Xapian::QueryParser, [158](#)
- FLAG\_PHRASE
  - Xapian::QueryParser, [158](#)
- FLAG\_PURE\_NOT
  - Xapian::QueryParser, [158](#)
- FLAG\_SPELLING
  - Xapian::TermGenerator, [184](#)
- FLAG\_SPELLING\_CORRECTION
  - Xapian::QueryParser, [158](#)
- FLAG\_SYNONYM
  - Xapian::QueryParser, [158](#)
- FLAG\_WILDCARD
  - Xapian::QueryParser, [158](#)
- flags
  - Xapian::TermGenerator, [184](#)
- flush
  - Xapian::WritableDatabase, [219](#)
- get\_available\_languages
  - Xapian::Stem, [178](#)
- get\_avlength
  - Xapian::Database, [52](#)
- get\_category
  - Xapian::Unicode, [36](#)
- get\_collapse\_count
  - Xapian::MSetIterator, [132](#)
- get\_collapse\_key
  - Xapian::MSetIterator, [132](#)
- get\_collection\_freq
  - Xapian::Database, [52](#)
- get\_context
  - Xapian::Error, [96](#)
- get\_corrected\_query\_string
  - Xapian::QueryParser, [161](#)
- get\_data
  - Xapian::Document, [78](#)
- get\_default\_op
  - Xapian::QueryParser, [161](#)
- get\_description
  - Xapian::Database, [52](#)
  - Xapian::Document, [78](#)
  - Xapian::Enquire, [85](#)
  - Xapian::Error, [96](#)
  - Xapian::ESet, [102](#)
  - Xapian::ESetIterator, [105](#)
  - Xapian::MSet, [126](#)
  - Xapian::MSetIterator, [133](#)
  - Xapian::PositionIterator, [144](#)
  - Xapian::PostingIterator, [148](#)
  - Xapian::Query, [154](#)
  - Xapian::QueryParser, [161](#)
  - Xapian::RSet, [170](#)
  - Xapian::SimpleStopper, [173](#)
  - Xapian::Stem, [178](#)
  - Xapian::Stopper, [179](#)
  - Xapian::TermGenerator, [185](#)
  - Xapian::TermIterator, [190](#)
  - Xapian::ValueIterator, [205](#)
  - Xapian::WritableDatabase, [220](#)
- get\_doccount
  - Xapian::Database, [53](#)
- get\_docid
  - Xapian::Document, [78](#)
- get\_doclength
  - Xapian::Database, [53](#)
  - Xapian::PostingIterator, [148](#)
- get\_document
  - Xapian::Database, [53](#)
  - Xapian::MSetIterator, [133](#)
  - Xapian::TermGenerator, [185](#)

- get\_ebound
  - Xapian::ESet, 102
- get\_errno
  - Xapian::Error, 96
- get\_error\_string
  - Xapian::Error, 96
- get\_eset
  - Xapian::Enquire, 85, 86
- get\_firstitem
  - Xapian::MSet, 127
- get\_lastdocid
  - Xapian::Database, 53
- get\_length
  - Xapian::Query, 155
- get\_matches\_estimated
  - Xapian::MSet, 127
- get\_matches\_lower\_bound
  - Xapian::MSet, 127
- get\_matches\_upper\_bound
  - Xapian::MSet, 127
- get\_matching\_terms\_begin
  - Xapian::Enquire, 86, 87
- get\_matching\_terms\_end
  - Xapian::Enquire, 87, 88
- get\_max\_attained
  - Xapian::MSet, 127
- get\_max\_possible
  - Xapian::MSet, 128
- get\_maxextra
  - Xapian::BM25Weight, 43
  - Xapian::BoolWeight, 46
  - Xapian::TradWeight, 194
  - Xapian::Weight, 210
- get\_maxpart
  - Xapian::BM25Weight, 43
  - Xapian::BoolWeight, 46
  - Xapian::TradWeight, 194
  - Xapian::Weight, 210
- get\_metadata
  - Xapian::Database, 53
- get\_mset
  - Xapian::Enquire, 88
- get\_msg
  - Xapian::Error, 96
- get\_percent
  - Xapian::MSetIterator, 133
- get\_query
  - Xapian::Enquire, 89
- get\_rank
  - Xapian::MSetIterator, 133
- get\_spelling\_suggestion
  - Xapian::Database, 54
- get\_sumextra
  - Xapian::BM25Weight, 43
  - Xapian::BoolWeight, 46
  - Xapian::TradWeight, 194
  - Xapian::Weight, 211
- get\_sumpart
  - Xapian::BM25Weight, 43
  - Xapian::BoolWeight, 46
  - Xapian::TradWeight, 194
  - Xapian::Weight, 211
- get\_sumpart\_needs\_doclength
  - Xapian::BM25Weight, 44
  - Xapian::BoolWeight, 47
  - Xapian::TradWeight, 195
  - Xapian::Weight, 211
- get\_termfreq
  - Xapian::Database, 54
  - Xapian::MSet, 128
  - Xapian::TermIterator, 190
- get\_termpos
  - Xapian::TermGenerator, 185
- get\_terms\_begin
  - Xapian::Query, 155
- get\_terms\_end
  - Xapian::Query, 155
- get\_termweight
  - Xapian::MSet, 128
- get\_type
  - Xapian::Error, 96
- get\_value
  - Xapian::Document, 79
- get\_valueno
  - Xapian::ValueIterator, 205
- get\_wdf
  - Xapian::PostingIterator, 148
  - Xapian::TermIterator, 190
- get\_weight
  - Xapian::ESetIterator, 105
  - Xapian::MSetIterator, 134
- has\_positions
  - Xapian::Database, 54
- include\_query\_terms
  - Xapian::Enquire, 93
- increase\_termpos
  - Xapian::TermGenerator, 185
- index\_text

- Xapian::TermGenerator, [185](#), [186](#)
- index\_text\_without\_positions
  - Xapian::TermGenerator, [186](#)
- InternalError
  - Xapian::InternalError, [114](#)
- InvalidArgumentError
  - Xapian::InvalidArgumentError, [116](#)
- InvalidOperationError
  - Xapian::InvalidOperationError, [118](#)
- is\_currency
  - Xapian::Unicode, [36](#)
- is\_whitespace
  - Xapian::Unicode, [37](#)
- is\_wordchar
  - Xapian::Unicode, [37](#)
- iterator
  - Xapian::MSet, [124](#)
- iterator\_category
  - Xapian::ESetIterator, [104](#)
  - Xapian::MSetIterator, [131](#)
  - Xapian::PostingIterator, [147](#)
  - Xapian::TermIterator, [189](#)
  - Xapian::Utf8Iterator, [199](#)
  - Xapian::ValueIterator, [204](#)
- keep\_alive
  - Xapian::Database, [54](#)
- left
  - Xapian::Utf8Iterator, [201](#)
- major\_version
  - Xapian, [22](#)
- MatchAll
  - Xapian::Query, [155](#)
- MatchNothing
  - Xapian::Query, [155](#)
- max\_size
  - Xapian::ESet, [102](#)
  - Xapian::MSet, [128](#)
- metadata\_keys\_begin
  - Xapian::Database, [54](#)
- metadata\_keys\_end
  - Xapian::Database, [55](#)
- minor\_version
  - Xapian, [22](#)
- MSet
  - Xapian::MSet, [125](#)
- MSetIterator
  - Xapian::MSetIterator, [132](#)
- name
  - Xapian::BM25Weight, [44](#)
  - Xapian::BoolWeight, [47](#)
  - Xapian::TradWeight, [195](#)
  - Xapian::Weight, [211](#)
- NetworkError
  - Xapian::NetworkError, [138](#)
- NetworkTimeoutError
  - Xapian::NetworkTimeoutError, [140](#)
- nonascii\_to\_utf8
  - Xapian::Unicode, [37](#)
- NumberValueRangeProcessor
  - Xapian::NumberValueRangeProcessor, [141](#), [142](#)
- op
  - Xapian::Query, [152](#)
- OP\_AND
  - Xapian::Query, [152](#)
- OP\_AND\_MAYBE
  - Xapian::Query, [152](#)
- OP\_AND\_NOT
  - Xapian::Query, [152](#)
- OP\_ELITE\_SET
  - Xapian::Query, [152](#)
- OP\_FILTER
  - Xapian::Query, [152](#)
- OP\_NEAR
  - Xapian::Query, [152](#)
- OP\_OR
  - Xapian::Query, [152](#)
- OP\_PHRASE
  - Xapian::Query, [152](#)
- OP\_SCALE\_WEIGHT
  - Xapian::Query, [152](#)
- OP\_VALUE\_GE
  - Xapian::Query, [152](#)
- OP\_VALUE\_LE
  - Xapian::Query, [152](#)
- OP\_VALUE\_RANGE
  - Xapian::Query, [152](#)
- OP\_XOR
  - Xapian::Query, [152](#)
- open
  - Xapian::Flint, [27](#)
  - Xapian::InMemory, [29](#)
  - Xapian::Quartz, [30](#)
  - Xapian::Remote, [32](#), [33](#)
- open\_stub
  - Xapian::Auto, [26](#)

- open\_writable
  - Xapian::Remote, 33
- operator\*
  - Xapian::ESetIterator, 105
  - Xapian::MSetIterator, 134
  - Xapian::PostingIterator, 148
  - Xapian::TermIterator, 190
  - Xapian::Utf8Iterator, 201
  - Xapian::ValueIterator, 205
- operator()
  - Xapian::DateValueRangeProcessor, 72
  - Xapian::ErrorHandler, 98
  - Xapian::ExpandDecider, 107
  - Xapian::ExpandDeciderAnd, 109
  - Xapian::ExpandDeciderFilterTerms, 111
  - Xapian::MatchDecider, 121
  - Xapian::MultiValueSorter, 136
  - Xapian::NumberValueRangeProcessor, 142
  - Xapian::SimpleStopper, 173
  - Xapian::Sorter, 174
  - Xapian::Stem, 178
  - Xapian::Stopper, 179
  - Xapian::StringValueRangeProcessor, 182
  - Xapian::ValueRangeProcessor, 207
- operator++
  - Xapian::ESetIterator, 105
  - Xapian::MSetIterator, 134
  - Xapian::Utf8Iterator, 201, 202
  - Xapian::ValueIterator, 205
- operator->
  - Xapian::ValueIterator, 205
- operator-
  - Xapian::ESetIterator, 105
  - Xapian::MSetIterator, 134
- operator=
  - Xapian::Database, 55
  - Xapian::Document, 79
  - Xapian::Enquire, 89
  - Xapian::ESet, 102
  - Xapian::ESetIterator, 105
  - Xapian::MSet, 128
  - Xapian::MSetIterator, 134
  - Xapian::PositionIterator, 144
  - Xapian::PostingIterator, 148
  - Xapian::Query, 155
  - Xapian::QueryParser, 161
  - Xapian::RSet, 170
  - Xapian::Stem, 178
  - Xapian::TermGenerator, 186
  - Xapian::TermIterator, 190
  - Xapian::ValueIterator, 205
  - Xapian::WritableDatabase, 220
- operator==
  - Xapian, 23
  - Xapian::ESetIterator, 106
  - Xapian::MSetIterator, 134
  - Xapian::PositionIterator, 144
  - Xapian::PostingIterator, 149
  - Xapian::Utf8Iterator, 202
  - Xapian::ValueIterator, 206
- parse\_query
  - Xapian::QueryParser, 161
- percent
  - Xapian, 21
- pointer
  - Xapian::ESetIterator, 104
  - Xapian::MSet, 125
  - Xapian::MSetIterator, 132
  - Xapian::PostingIterator, 147
  - Xapian::TermIterator, 189
  - Xapian::Utf8Iterator, 199
  - Xapian::ValueIterator, 204
- PositionIterator
  - Xapian::PositionIterator, 144
- positionlist\_begin
  - Xapian::Database, 55
  - Xapian::PostingIterator, 149
  - Xapian::TermIterator, 191
- positionlist\_count
  - Xapian::TermIterator, 191
- positionlist\_end
  - Xapian::Database, 55
  - Xapian::PostingIterator, 149
  - Xapian::TermIterator, 191
- PostingIterator
  - Xapian::PostingIterator, 148
- postlist\_begin
  - Xapian::Database, 55
- postlist\_end
  - Xapian::Database, 56
- Query
  - Xapian::Query, 152–154
- QueryParser
  - Xapian::QueryParser, 159

- QueryParserError
  - Xapian::QueryParserError, [164](#)
- RangeError
  - Xapian::RangeError, [166](#)
- raw
  - Xapian::Utf8Iterator, [202](#)
- reference
  - Xapian::ESetIterator, [104](#)
  - Xapian::MSet, [125](#)
  - Xapian::MSetIterator, [132](#)
  - Xapian::PostingIterator, [147](#)
  - Xapian::TermIterator, [189](#)
  - Xapian::Utf8Iterator, [199](#)
  - Xapian::ValueIterator, [204](#)
- register\_match\_decider
  - Xapian::Enquire, [89](#)
- remove\_document
  - Xapian::RSet, [170](#)
- remove\_posting
  - Xapian::Document, [79](#)
- remove\_spelling
  - Xapian::WritableDatabase, [220](#)
- remove\_synonym
  - Xapian::WritableDatabase, [220](#)
- remove\_term
  - Xapian::Document, [79](#)
- remove\_value
  - Xapian::Document, [80](#)
- reopen
  - Xapian::Database, [56](#)
- replace\_document
  - Xapian::WritableDatabase, [220](#), [221](#)
- revision
  - Xapian, [23](#)
- RSet
  - Xapian::RSet, [169](#)
- serialise
  - Xapian::BM25Weight, [44](#)
  - Xapian::BoolWeight, [47](#)
  - Xapian::TradWeight, [195](#)
  - Xapian::Weight, [212](#)
- set\_collapse\_key
  - Xapian::Enquire, [89](#)
- set\_cutoff
  - Xapian::Enquire, [90](#)
- set\_data
  - Xapian::Document, [80](#)
- set\_database
  - Xapian::QueryParser, [162](#)
  - Xapian::TermGenerator, [186](#)
- set\_default\_op
  - Xapian::QueryParser, [162](#)
- set\_docid\_order
  - Xapian::Enquire, [90](#)
- set\_document
  - Xapian::TermGenerator, [186](#)
- set\_flags
  - Xapian::TermGenerator, [186](#)
- set\_metadata
  - Xapian::WritableDatabase, [222](#)
- set\_query
  - Xapian::Enquire, [91](#)
- set\_sort\_by\_key
  - Xapian::Enquire, [91](#)
- set\_sort\_by\_key\_then\_relevance
  - Xapian::Enquire, [91](#)
- set\_sort\_by\_relevance
  - Xapian::Enquire, [91](#)
- set\_sort\_by\_relevance\_then\_key
  - Xapian::Enquire, [92](#)
- set\_sort\_by\_relevance\_then\_value
  - Xapian::Enquire, [92](#)
- set\_sort\_by\_value
  - Xapian::Enquire, [92](#)
- set\_sort\_by\_value\_then\_relevance
  - Xapian::Enquire, [93](#)
- set\_stemmer
  - Xapian::QueryParser, [162](#)
  - Xapian::TermGenerator, [187](#)
- set\_stemming\_strategy
  - Xapian::QueryParser, [162](#)
- set\_stopper
  - Xapian::QueryParser, [163](#)
  - Xapian::TermGenerator, [187](#)
- set\_termpos
  - Xapian::TermGenerator, [187](#)
- set\_weighting\_scheme
  - Xapian::Enquire, [93](#)
- SimpleStopper
  - Xapian::SimpleStopper, [173](#)
- size
  - Xapian::ESet, [102](#)
  - Xapian::MSet, [129](#)
  - Xapian::RSet, [170](#)
- size\_type
  - Xapian::MSet, [125](#)
- skip\_to
  - Xapian::PositionIterator, [144](#)

- Xapian::PostingIterator, 149
- Xapian::TermIterator, 191
- sortable\_serialise
  - Xapian, 24
- sortable\_unserialise
  - Xapian, 24
- spellings\_begin
  - Xapian::Database, 56
- spellings\_end
  - Xapian::Database, 56
- Stem
  - Xapian::Stem, 176, 177
- stoplist\_begin
  - Xapian::QueryParser, 163
- StringValueRangeProcessor
  - Xapian::StringValueRangeProcessor, 181
- swap
  - Xapian::ESet, 102
  - Xapian::MSet, 129
- synonym\_keys\_begin
  - Xapian::Database, 56
- synonym\_keys\_end
  - Xapian::Database, 56
- synonyms\_begin
  - Xapian::Database, 56
- synonyms\_end
  - Xapian::Database, 57
- term\_exists
  - Xapian::Database, 57
- termcount
  - Xapian, 21
- termcount\_diff
  - Xapian, 21
- TermGenerator
  - Xapian::TermGenerator, 185
- TermIterator
  - Xapian::TermIterator, 190
- termlist\_begin
  - Xapian::Database, 57
  - Xapian::Document, 80
- termlist\_count
  - Xapian::Document, 80
- termlist\_end
  - Xapian::Database, 57
  - Xapian::Document, 80
- termpos
  - Xapian, 21
- termpos\_diff
  - Xapian, 21
- timeout
  - Xapian, 21
- to\_utf8
  - Xapian::Unicode, 37
- tolower
  - Xapian::Unicode, 37
- toupper
  - Xapian::Unicode, 38
- TradWeight
  - Xapian::TradWeight, 193
- UnimplementedError
  - Xapian::UnimplementedError, 196
- unserialise
  - Xapian::BM25Weight, 44
  - Xapian::BoolWeight, 47
  - Xapian::TradWeight, 195
  - Xapian::Weight, 212
- unstem\_begin
  - Xapian::QueryParser, 163
- use\_exact\_termfreq
  - Xapian::Enquire, 93
- Utf8Iterator
  - Xapian::Utf8Iterator, 200
- value\_type
  - Xapian::ESetIterator, 104
  - Xapian::MSet, 125
  - Xapian::MSetIterator, 132
  - Xapian::PostingIterator, 147
  - Xapian::TermIterator, 189
  - Xapian::Utf8Iterator, 199
  - Xapian::ValueIterator, 204
- ValueIterator
  - Xapian::ValueIterator, 205
- valueno
  - Xapian, 21
- valueno\_diff
  - Xapian, 21
- values\_begin
  - Xapian::Document, 80
- values\_count
  - Xapian::Document, 80
- values\_end
  - Xapian::Document, 80
- version.h
  - XAPIAN\_HAS\_FLINT\_BACKEND, 228



- XAPIAN\_HAS\_INMEMORY\_-  
  BACKEND, 228
- XAPIAN\_HAS\_QUARTZ\_-  
  BACKEND, 229
- XAPIAN\_HAS\_REMOTE\_-  
  BACKEND, 229
- XAPIAN\_MAJOR\_VERSION, 229
- XAPIAN\_MINOR\_VERSION, 229
- XAPIAN\_REVISION, 229
- XAPIAN\_VERSION, 229
- version\_string  
  Xapian, 24
- weight  
  Xapian, 22
- WritableDatabase  
  Xapian::WritableDatabase, 215
- WritableDatabaseopen  
  Xapian::Quartz, 30
- Xapian, 13
  - BAD\_VALUENO, 25
  - DB\_CREATE, 25
  - DB\_CREATE\_OR\_OPEN, 25
  - DB\_CREATE\_OR\_OVERWRITE,  
  25
  - DB\_OPEN, 25
  - doccount, 20
  - doccount\_diff, 20
  - docid, 20
  - doclength, 20
  - major\_version, 22
  - minor\_version, 22
  - operator==, 23
  - percent, 21
  - revision, 23
  - sortable\_serialise, 24
  - sortable\_unserialise, 24
  - termcount, 21
  - termcount\_diff, 21
  - termpos, 21
  - termpos\_diff, 21
  - timeout, 21
  - valueno, 21
  - valueno\_diff, 21
  - version\_string, 24
  - weight, 22
  - xapian\_major\_version, 24
  - xapian\_minor\_version, 25
  - xapian\_revision, 25
  - xapian\_version\_string, 25
- xapian.h, 230
- xapian/database.h, 231
- xapian/dbfactory.h, 232
- xapian/document.h, 234
- xapian/enquire.h, 235
- xapian/error.h, 225
- xapian/errorhandler.h, 237
- xapian/expanddecider.h, 238
- xapian/positioniterator.h, 239
- xapian/postingiterator.h, 240
- xapian/query.h, 241
- xapian/queryparser.h, 242
- xapian/sorter.h, 244
- xapian/stem.h, 245
- xapian/termgenerator.h, 246
- xapian/termiterator.h, 247
- xapian/types.h, 248
- xapian/unicode.h, 250
- xapian/valueiterator.h, 252
- xapian/version.h, 228
- Xapian::AssertionError, 39
  - AssertionError, 40
- Xapian::Auto, 26
  - open\_stub, 26
- Xapian::BM25Weight, 41
  - BM25Weight, 42
  - clone, 43
  - get\_maxextra, 43
  - get\_maxpart, 43
  - get\_sumextra, 43
  - get\_sumpart, 43
  - get\_sumpart\_needs\_doclength, 44
  - name, 44
  - serialise, 44
  - unserialise, 44
- Xapian::BoolWeight, 45
  - clone, 46
  - get\_maxextra, 46
  - get\_maxpart, 46
  - get\_sumextra, 46
  - get\_sumpart, 46
  - get\_sumpart\_needs\_doclength, 47
  - name, 47
  - serialise, 47
  - unserialise, 47
- Xapian::Database, 48
  - ~Database, 51
  - add\_database, 51
  - allterms\_begin, 52

- allterms\_end, 52
- Database, 51
- get\_avlength, 52
- get\_collection\_freq, 52
- get\_description, 52
- get\_doccount, 53
- get\_doclength, 53
- get\_document, 53
- get\_lastdocid, 53
- get\_metadata, 53
- get\_spelling\_suggestion, 54
- get\_termfreq, 54
- has\_positions, 54
- keep\_alive, 54
- metadata\_keys\_begin, 54
- metadata\_keys\_end, 55
- operator=, 55
- positionlist\_begin, 55
- positionlist\_end, 55
- postlist\_begin, 55
- postlist\_end, 56
- reopen, 56
- spellings\_begin, 56
- spellings\_end, 56
- synonym\_keys\_begin, 56
- synonym\_keys\_end, 56
- synonyms\_begin, 56
- synonyms\_end, 57
- term\_exists, 57
- termlist\_begin, 57
- termlist\_end, 57
- Xapian::DatabaseCorruptError, 58
  - DatabaseCorruptError, 59
- Xapian::DatabaseCreateError, 60
  - DatabaseCreateError, 61
- Xapian::DatabaseError, 62
  - DatabaseError, 62
- Xapian::DatabaseLockError, 63
  - DatabaseLockError, 64
- Xapian::DatabaseModifiedError, 65
  - DatabaseModifiedError, 66
- Xapian::DatabaseOpeningError, 67
  - DatabaseOpeningError, 68
- Xapian::DatabaseVersionError, 69
  - DatabaseVersionError, 70
- Xapian::DateValueRangeProcessor, 71
  - DateValueRangeProcessor, 71
  - operator(), 72
- Xapian::DocNotFoundError, 73
  - DocNotFoundError, 73
- Xapian::Document, 75
  - ~Document, 77
  - add\_boolean\_term, 77
  - add\_posting, 77
  - add\_term, 78
  - add\_value, 78
  - clear\_terms, 78
  - clear\_values, 78
  - Document, 77
  - get\_data, 78
  - get\_description, 78
  - get\_docid, 78
  - get\_value, 79
  - operator=, 79
  - remove\_posting, 79
  - remove\_term, 79
  - remove\_value, 80
  - set\_data, 80
  - termlist\_begin, 80
  - termlist\_count, 80
  - termlist\_end, 80
  - values\_begin, 80
  - values\_count, 80
  - values\_end, 80
- Xapian::Enquire, 82
  - ~Enquire, 85
  - Enquire, 85
  - get\_description, 85
  - get\_eset, 85, 86
  - get\_matching\_terms\_begin, 86, 87
  - get\_matching\_terms\_end, 87, 88
  - get\_mset, 88
  - get\_query, 89
  - include\_query\_terms, 93
  - operator=, 89
  - register\_match\_decider, 89
  - set\_collapse\_key, 89
  - set\_cutoff, 90
  - set\_docid\_order, 90
  - set\_query, 91
  - set\_sort\_by\_key, 91
  - set\_sort\_by\_key\_then\_relevance, 91
  - set\_sort\_by\_relevance, 91
  - set\_sort\_by\_relevance\_then\_key, 92
  - set\_sort\_by\_relevance\_then\_value, 92
  - set\_sort\_by\_value, 92
  - set\_sort\_by\_value\_then\_relevance, 93
  - set\_weighting\_scheme, 93

- use\_exact\_termfreq, 93
- Xapian::Error, 95
  - get\_context, 96
  - get\_description, 96
  - get\_errno, 96
  - get\_error\_string, 96
  - get\_msg, 96
  - get\_type, 96
- Xapian::ErrorHandler, 98
  - ~ErrorHandler, 98
  - ErrorHandler, 98
  - operator(), 98
- Xapian::ESet, 100
  - ~ESet, 101
  - back, 101
  - begin, 101
  - empty, 101
  - end, 101
  - ESet, 101
  - get\_description, 102
  - get\_ebound, 102
  - max\_size, 102
  - operator=, 102
  - size, 102
  - swap, 102
- Xapian::ESetIterator, 103
  - difference\_type, 104
  - ESetIterator, 105
  - get\_description, 105
  - get\_weight, 105
  - iterator\_category, 104
  - operator\*, 105
  - operator++, 105
  - operator--, 105
  - operator=, 105
  - operator==, 106
  - pointer, 104
  - reference, 104
  - value\_type, 104
- Xapian::ExpandDecider, 107
  - ~ExpandDecider, 107
  - operator(), 107
- Xapian::ExpandDeciderAnd, 108
  - ExpandDeciderAnd, 108
  - operator(), 109
- Xapian::ExpandDeciderFilterTerms, 110
  - ExpandDeciderFilterTerms, 110
  - operator(), 111
- Xapian::FeatureUnavailableError, 112
  - FeatureUnavailableError, 112
- Xapian::Flint, 27
  - open, 27
- Xapian::InMemory, 29
  - open, 29
- Xapian::InternalError, 114
  - InternalError, 114
- Xapian::InvalidArgumentError, 116
  - InvalidArgumentError, 116
- Xapian::InvalidOperationError, 118
  - InvalidOperationError, 118
- Xapian::LogicError, 120
- Xapian::MatchDecider, 121
  - ~MatchDecider, 121
  - operator(), 121
- Xapian::MSet, 122
  - ~MSet, 125
  - back, 125
  - begin, 125
  - const\_iterator, 124
  - const\_reference, 124
  - convert\_to\_percent, 125, 126
  - difference\_type, 124
  - empty, 126
  - end, 126
  - fetch, 126
  - get\_description, 126
  - get\_firstitem, 127
  - get\_matches\_estimated, 127
  - get\_matches\_lower\_bound, 127
  - get\_matches\_upper\_bound, 127
  - get\_max\_attained, 127
  - get\_max\_possible, 128
  - get\_termfreq, 128
  - get\_termweight, 128
  - iterator, 124
  - max\_size, 128
  - MSet, 125
  - operator=, 128
  - pointer, 125
  - reference, 125
  - size, 129
  - size\_type, 125
  - swap, 129
  - value\_type, 125
- Xapian::MSetIterator, 130
  - difference\_type, 131
  - get\_collapse\_count, 132
  - get\_collapse\_key, 132
  - get\_description, 133
  - get\_document, 133

- get\_percent, 133
- get\_rank, 133
- get\_weight, 134
- iterator\_category, 131
- MSetIterator, 132
- operator\*, 134
- operator++, 134
- operator-, 134
- operator=, 134
- operator==, 134
- pointer, 132
- reference, 132
- value\_type, 132
- Xapian::MultiValueSorter, 136
  - operator(), 136
- Xapian::NetworkError, 137
  - NetworkError, 138
- Xapian::NetworkTimeoutError, 139
  - NetworkTimeoutError, 140
- Xapian::NumberValueRangeProcessor, 141
  - NumberValueRangeProcessor, 141, 142
  - operator(), 142
- Xapian::PositionIterator, 143
  - ~PositionIterator, 144
  - get\_description, 144
  - operator=, 144
  - operator==, 144
  - PositionIterator, 144
  - skip\_to, 144
- Xapian::PostingIterator, 146
  - ~PostingIterator, 148
  - difference\_type, 147
  - get\_description, 148
  - get\_doclength, 148
  - get\_wdf, 148
  - iterator\_category, 147
  - operator\*, 148
  - operator=, 148
  - operator==, 149
  - pointer, 147
  - positionlist\_begin, 149
  - positionlist\_end, 149
  - PostingIterator, 148
  - reference, 147
  - skip\_to, 149
  - value\_type, 147
- Xapian::Quartz, 30
  - open, 30
  - WritableDatabaseopen, 30
- Xapian::Query, 150
  - ~Query, 153
  - empty, 154
  - get\_description, 154
  - get\_length, 155
  - get\_terms\_begin, 155
  - get\_terms\_end, 155
  - MatchAll, 155
  - MatchNothing, 155
  - op, 152
  - OP\_AND, 152
  - OP\_AND\_MAYBE, 152
  - OP\_AND\_NOT, 152
  - OP\_ELITE\_SET, 152
  - OP\_FILTER, 152
  - OP\_NEAR, 152
  - OP\_OR, 152
  - OP\_PHRASE, 152
  - OP\_SCALE\_WEIGHT, 152
  - OP\_VALUE\_GE, 152
  - OP\_VALUE\_LE, 152
  - OP\_VALUE\_RANGE, 152
  - OP\_XOR, 152
  - operator=, 155
  - Query, 152–154
- Xapian::QueryParser, 156
  - ~QueryParser, 159
  - add\_boolean\_prefix, 159
  - add\_prefix, 160
  - add\_valuerangeprocessor, 160
  - feature\_flag, 157
  - FLAG\_AUTO\_MULTIWORD\_-SYNONYMS, 158
  - FLAG\_AUTO\_SYNONYMS, 158
  - FLAG\_BOOLEAN, 158
  - FLAG\_BOOLEAN\_ANY\_CASE, 158
  - FLAG\_DEFAULT, 158
  - FLAG\_LOVEHATE, 158
  - FLAG\_PARTIAL, 158
  - FLAG\_PHRASE, 158
  - FLAG\_PURE\_NOT, 158
  - FLAG\_SPELLING\_-CORRECTION, 158
  - FLAG\_SYNONYM, 158
  - FLAG\_WILDCARD, 158
  - get\_corrected\_query\_string, 161
  - get\_default\_op, 161
  - get\_description, 161

- operator=, 161
- parse\_query, 161
- QueryParser, 159
- set\_database, 162
- set\_default\_op, 162
- set\_stemmer, 162
- set\_stemming\_strategy, 162
- set\_stopper, 163
- stoplist\_begin, 163
- unstem\_begin, 163
- Xapian::QueryParserError, 164
  - QueryParserError, 164
- Xapian::RangeError, 166
  - RangeError, 166
- Xapian::Remote, 32
  - open, 32, 33
  - open\_writable, 33
- Xapian::RSet, 168
  - ~RSet, 169
  - add\_document, 169
  - contains, 169, 170
  - empty, 170
  - get\_description, 170
  - operator=, 170
  - remove\_document, 170
  - RSet, 169
  - size, 170
- Xapian::RuntimeError, 171
- Xapian::SimpleStopper, 172
  - ~SimpleStopper, 173
  - add, 173
  - get\_description, 173
  - operator(), 173
  - SimpleStopper, 173
- Xapian::Sorter, 174
  - ~Sorter, 174
  - operator(), 174
- Xapian::Stem, 176
  - ~Stem, 177
  - get\_available\_languages, 178
  - get\_description, 178
  - operator(), 178
  - operator=, 178
  - Stem, 176, 177
- Xapian::Stopper, 179
  - ~Stopper, 179
  - get\_description, 179
  - operator(), 179
- Xapian::StringValueRangeProcessor, 181
  - operator(), 182
  - StringValueRangeProcessor, 181
- Xapian::TermGenerator, 183
  - ~TermGenerator, 185
  - FLAG\_SPELLING, 184
  - flags, 184
  - get\_description, 185
  - get\_document, 185
  - get\_termpos, 185
  - increase\_termpos, 185
  - index\_text, 185, 186
  - index\_text\_without\_positions, 186
  - operator=, 186
  - set\_database, 186
  - set\_document, 186
  - set\_flags, 186
  - set\_stemmer, 187
  - set\_stopper, 187
  - set\_termpos, 187
  - TermGenerator, 185
- Xapian::TermIterator, 188
  - ~TermIterator, 190
  - difference\_type, 189
  - get\_description, 190
  - get\_termfreq, 190
  - get\_wdf, 190
  - iterator\_category, 189
  - operator\*, 190
  - operator=, 190
  - pointer, 189
  - positionlist\_begin, 191
  - positionlist\_count, 191
  - positionlist\_end, 191
  - reference, 189
  - skip\_to, 191
  - TermIterator, 190
  - value\_type, 189
- Xapian::TradWeight, 192
  - clone, 193
  - get\_maxextra, 194
  - get\_maxpart, 194
  - get\_sumextra, 194
  - get\_sumpart, 194
  - get\_sumpart\_needs\_doclength, 195
  - name, 195
  - serialise, 195
  - TradWeight, 193
  - unserialise, 195
- Xapian::Unicode, 35
  - append\_utf8, 36
  - category, 36

- get\_category, 36
  - is\_currency, 36
  - is\_whitespace, 37
  - is\_wordchar, 37
  - nonascii\_to\_utf8, 37
  - to\_utf8, 37
  - tolower, 37
  - toupper, 38
- Xapian::UnimplementedError, 196
  - UnimplementedError, 196
- Xapian::Utf8Iterator, 198
  - assign, 201
  - difference\_type, 199
  - iterator\_category, 199
  - left, 201
  - operator\*, 201
  - operator++, 201, 202
  - operator==, 202
  - pointer, 199
  - raw, 202
  - reference, 199
  - Utf8Iterator, 200
  - value\_type, 199
- Xapian::ValueIterator, 203
  - difference\_type, 204
  - get\_description, 205
  - get\_valueno, 205
  - iterator\_category, 204
  - operator\*, 205
  - operator++, 205
  - operator->, 205
  - operator=, 205
  - operator==, 206
  - pointer, 204
  - reference, 204
  - value\_type, 204
  - ValueIterator, 205
- Xapian::ValueRangeProcessor, 207
  - ~ValueRangeProcessor, 207
  - operator(), 207
- Xapian::Weight, 209
  - create, 210
  - get\_maxextra, 210
  - get\_maxpart, 210
  - get\_sumextra, 211
  - get\_sumpart, 211
  - get\_sumpart\_needs\_doclength, 211
  - name, 211
  - serialise, 212
  - unserialise, 212
- Xapian::WritableDatabase, 213
  - ~WritableDatabase, 215
  - add\_document, 216
  - add\_spelling, 216
  - add\_synonym, 216
  - begin\_transaction, 217
  - cancel\_transaction, 217
  - clear\_synonyms, 218
  - commit\_transaction, 218
  - delete\_document, 218, 219
  - flush, 219
  - get\_description, 220
  - operator=, 220
  - remove\_spelling, 220
  - remove\_synonym, 220
  - replace\_document, 220, 221
  - set\_metadata, 222
  - WritableDatabase, 215
- XAPIAN\_HAS\_FLINT\_BACKEND
  - version.h, 228
- XAPIAN\_HAS\_INMEMORY\_ -  
BACKEND
  - version.h, 228
- XAPIAN\_HAS\_QUARTZ\_BACKEND
  - version.h, 229
- XAPIAN\_HAS\_REMOTE\_BACKEND
  - version.h, 229
- XAPIAN\_MAJOR\_VERSION
  - version.h, 229
- xapian\_major\_version
  - Xapian, 24
- XAPIAN\_MINOR\_VERSION
  - version.h, 229
- xapian\_minor\_version
  - Xapian, 25
- XAPIAN\_REVISION
  - version.h, 229
- xapian\_revision
  - Xapian, 25
- XAPIAN\_VERSION
  - version.h, 229
- xapian\_version\_string
  - Xapian, 25