

# xapian-core Reference Manual

## 1.0.10

Generated by Doxygen 1.5.2

Tue Dec 23 08:11:23 2008



# Contents

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>xapian-core Namespace Index</b>                        | <b>1</b>  |
| 1.1      | xapian-core Namespace List . . . . .                      | 1         |
| <b>2</b> | <b>xapian-core Hierarchical Index</b>                     | <b>3</b>  |
| 2.1      | xapian-core Class Hierarchy . . . . .                     | 3         |
| <b>3</b> | <b>xapian-core Class Index</b>                            | <b>5</b>  |
| 3.1      | xapian-core Class List . . . . .                          | 5         |
| <b>4</b> | <b>xapian-core File Index</b>                             | <b>7</b>  |
| 4.1      | xapian-core File List . . . . .                           | 7         |
| <b>5</b> | <b>xapian-core Page Index</b>                             | <b>9</b>  |
| 5.1      | xapian-core Related Pages . . . . .                       | 9         |
| <b>6</b> | <b>xapian-core Namespace Documentation</b>                | <b>11</b> |
| 6.1      | Xapian Namespace Reference . . . . .                      | 11        |
| <b>7</b> | <b>xapian-core Class Documentation</b>                    | <b>23</b> |
| 7.1      | Xapian::BM25Weight Class Reference . . . . .              | 23        |
| 7.2      | Xapian::BoolWeight Class Reference . . . . .              | 28        |
| 7.3      | Xapian::CategorySelectMatchSpy Class Reference . . . . .  | 32        |
| 7.4      | Xapian::Database Class Reference . . . . .                | 35        |
| 7.5      | Xapian::DateValueRangeProcessor Class Reference . . . . . | 45        |
| 7.6      | Xapian::Document Class Reference . . . . .                | 47        |
| 7.7      | Xapian::Enquire Class Reference . . . . .                 | 53        |
| 7.8      | Xapian::ErrorHandler Class Reference . . . . .            | 65        |
| 7.9      | Xapian::ESet Class Reference . . . . .                    | 67        |

|  |     |
|--|-----|
| 7.10 Xapian::ESetIterator Class Reference . . . . .              | 70  |
| 7.11 Xapian::ExpandDecider Class Reference . . . . .             | 73  |
| 7.12 Xapian::ExpandDeciderAnd Class Reference . . . . .          | 74  |
| 7.13 Xapian::ExpandDeciderFilterTerms Class Reference . . . . .  | 76  |
| 7.14 Xapian::MatchDecider Class Reference . . . . .              | 78  |
| 7.15 Xapian::MSet Class Reference . . . . .                      | 80  |
| 7.16 Xapian::MSetIterator Class Reference . . . . .              | 87  |
| 7.17 Xapian::MultipleMatchDecider Class Reference . . . . .      | 92  |
| 7.18 Xapian::MultiValueSorter Class Reference . . . . .          | 94  |
| 7.19 Xapian::NumberValueRangeProcessor Class Reference . . . . . | 96  |
| 7.20 Xapian::PositionIterator Class Reference . . . . .          | 99  |
| 7.21 Xapian::PostingIterator Class Reference . . . . .           | 101 |
| 7.22 Xapian::Query Class Reference . . . . .                     | 105 |
| 7.23 Xapian::QueryParser Class Reference . . . . .               | 115 |
| 7.24 Xapian::RSet Class Reference . . . . .                      | 122 |
| 7.25 Xapian::SimpleStopper Class Reference . . . . .             | 125 |
| 7.26 Xapian::Sorter Class Reference . . . . .                    | 127 |
| 7.27 Xapian::Stem Class Reference . . . . .                      | 128 |
| 7.28 Xapian::Stopper Class Reference . . . . .                   | 131 |
| 7.29 Xapian::StringAndFrequency Struct Reference . . . . .       | 133 |
| 7.30 Xapian::StringListSerialiser Class Reference . . . . .      | 134 |
| 7.31 Xapian::StringListUnserialiser Class Reference . . . . .    | 136 |
| 7.32 Xapian::StringValueRangeProcessor Class Reference . . . . . | 139 |
| 7.33 Xapian::TermCountMatchSpy Class Reference . . . . .         | 141 |
| 7.34 Xapian::TermGenerator Class Reference . . . . .             | 145 |
| 7.35 Xapian::TermIterator Class Reference . . . . .              | 150 |
| 7.36 Xapian::TradWeight Class Reference . . . . .                | 154 |
| 7.37 Xapian::Utf8Iterator Class Reference . . . . .              | 158 |
| 7.38 Xapian::ValueCountMatchSpy Class Reference . . . . .        | 163 |
| 7.39 Xapian::ValueIterator Class Reference . . . . .             | 167 |
| 7.40 Xapian::ValueRangeProcessor Struct Reference . . . . .      | 170 |
| 7.41 Xapian::Weight Class Reference . . . . .                    | 172 |
| 7.42 Xapian::WritableDatabase Class Reference . . . . .          | 176 |

|          |  |            |
|----------|--|------------|
| <b>8</b> | <b>xapian-core File Documentation</b>                      | <b>187</b> |
| 8.1      | include/xapian.h File Reference . . . . .                  | 187        |
| 8.2      | include/xapian/database.h File Reference . . . . .         | 189        |
| 8.3      | include/xapian/dbfactory.h File Reference . . . . .        | 191        |
| 8.4      | include/xapian/document.h File Reference . . . . .         | 193        |
| 8.5      | include/xapian/enquire.h File Reference . . . . .          | 194        |
| 8.6      | include/xapian/errorhandler.h File Reference . . . . .     | 196        |
| 8.7      | include/xapian/expanddecider.h File Reference . . . . .    | 197        |
| 8.8      | include/xapian/matchspy.h File Reference . . . . .         | 198        |
| 8.9      | include/xapian/positioniterator.h File Reference . . . . . | 200        |
| 8.10     | include/xapian/postingiterator.h File Reference . . . . .  | 201        |
| 8.11     | include/xapian/query.h File Reference . . . . .            | 202        |
| 8.12     | include/xapian/queryparser.h File Reference . . . . .      | 203        |
| 8.13     | include/xapian/sorter.h File Reference . . . . .           | 205        |
| 8.14     | include/xapian/stem.h File Reference . . . . .             | 206        |
| 8.15     | include/xapian/termgenerator.h File Reference . . . . .    | 207        |
| 8.16     | include/xapian/termiterator.h File Reference . . . . .     | 208        |
| 8.17     | include/xapian/types.h File Reference . . . . .            | 209        |
| 8.18     | include/xapian/unicode.h File Reference . . . . .          | 211        |
| 8.19     | include/xapian/valueiterator.h File Reference . . . . .    | 213        |
| <b>9</b> | <b>xapian-core Page Documentation</b>                      | <b>215</b> |
| 9.1      | Deprecated List . . . . .                                  | 215        |



# Chapter 1

## xapian-core Namespace Index

### 1.1 xapian-core Namespace List

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

[Xapian](#) (The [Xapian](#) library lives in the [Xapian](#) namespace ) . . . . . 11





## Chapter 2

# xapian-core Hierarchical Index

### 2.1 xapian-core Class Hierarchy

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

|  |     |
|--|-----|
| Xapian::Database . . . . .                 | 35  |
| Xapian::WritableDatabase . . . . .         | 176 |
| Xapian::Document . . . . .                 | 47  |
| Xapian::Enquire . . . . .                  | 53  |
| Xapian::ErrorHandler . . . . .             | 65  |
| Xapian::ESet . . . . .                     | 67  |
| Xapian::ESetIterator . . . . .             | 70  |
| Xapian::ExpandDecider . . . . .            | 73  |
| Xapian::ExpandDeciderAnd . . . . .         | 74  |
| Xapian::ExpandDeciderFilterTerms . . . . . | 76  |
| Xapian::MatchDecider . . . . .             | 78  |
| Xapian::MultipleMatchDecider . . . . .     | 92  |
| Xapian::TermCountMatchSpy . . . . .        | 141 |
| Xapian::ValueCountMatchSpy . . . . .       | 163 |
| Xapian::CategorySelectMatchSpy . . . . .   | 32  |
| Xapian::MSet . . . . .                     | 80  |
| Xapian::MSetIterator . . . . .             | 87  |
| Xapian::PositionIterator . . . . .         | 99  |
| Xapian::PostingIterator . . . . .          | 101 |
| Xapian::Query . . . . .                    | 105 |
| Xapian::QueryParser . . . . .              | 115 |
| Xapian::RSet . . . . .                     | 122 |
| Xapian::Sorter . . . . .                   | 127 |
| Xapian::MultiValueSorter . . . . .         | 94  |
| Xapian::Stem . . . . .                     | 128 |
| Xapian::Stopper . . . . .                  | 131 |
| Xapian::SimpleStopper . . . . .            | 125 |
| Xapian::StringAndFrequency . . . . .       | 133 |

|   |     |
|---|-----|
| Xapian::StringListSerialiser . . . . .      | 134 |
| Xapian::StringListUnserialiser . . . . .    | 136 |
| Xapian::TermGenerator . . . . .             | 145 |
| Xapian::TermIterator . . . . .              | 150 |
| Xapian::Utf8Iterator . . . . .              | 158 |
| Xapian::ValueIterator . . . . .             | 167 |
| Xapian::ValueRangeProcessor . . . . .       | 170 |
| Xapian::DateValueRangeProcessor . . . . .   | 45  |
| Xapian::NumberValueRangeProcessor . . . . . | 96  |
| Xapian::StringValueRangeProcessor . . . . . | 139 |
| Xapian::Weight . . . . .                    | 172 |
| Xapian::BM25Weight . . . . .                | 23  |
| Xapian::BoolWeight . . . . .                | 28  |
| Xapian::TradWeight . . . . .                | 154 |

## Chapter 3

# xapian-core Class Index

### 3.1 xapian-core Class List

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

|  |     |
|--|-----|
| <a href="#">Xapian::BM25Weight</a> (BM25 weighting scheme ) . . . . .  | 23  |
| <a href="#">Xapian::BoolWeight</a> (Boolean weighting scheme (everything gets 0) ) . . . .   | 28  |
| <a href="#">Xapian::CategorySelectMatchSpy</a> (MatchSpy for classifying matching documents by their values ) . . . . .                                    | 32  |
| <a href="#">Xapian::Database</a> (This class is used to access a database, or a group of databases ) . . . . .   | 35  |
| <a href="#">Xapian::DateValueRangeProcessor</a> (Handle a date range ) . . . . .   | 45  |
| <a href="#">Xapian::Document</a> (A document in the database - holds data, values, terms, and postings ) . . . . .   | 47  |
| <a href="#">Xapian::Enquire</a> (This class provides an interface to the information retrieval system for the purpose of searching ) . . . . .             | 53  |
| <a href="#">Xapian::ErrorHandler</a> (Decide if a Xapian::Error exception should be ignored ) . . . . .  | 65  |
| <a href="#">Xapian::ESet</a> (Class representing an ordered set of expand terms (an <a href="#">ESet</a> ) ) .   | 67  |
| <a href="#">Xapian::ESetIterator</a> (Iterate through terms in the <a href="#">ESet</a> ) . . . . .  | 70  |
| <a href="#">Xapian::ExpandDecider</a> (Virtual base class for expand decider functor ) . . .   | 73  |
| <a href="#">Xapian::ExpandDeciderAnd</a> ( <a href="#">ExpandDecider</a> subclass which rejects terms using two <a href="#">ExpandDeciders</a> ) . . . . . | 74  |
| <a href="#">Xapian::ExpandDeciderFilterTerms</a> ( <a href="#">ExpandDecider</a> subclass which rejects terms in a specified list ) . . . . .              | 76  |
| <a href="#">Xapian::MatchDecider</a> (Base class for matcher decision functor ) . . . . .  | 78  |
| <a href="#">Xapian::MSet</a> (A match set ( <a href="#">MSet</a> ) ) . . . . .   | 80  |
| <a href="#">Xapian::MSetIterator</a> (An iterator pointing to items in an <a href="#">MSet</a> ) . . . . .   | 87  |
| <a href="#">Xapian::MultipleMatchDecider</a> (Class which applies several match deciders in turn ) . . . . .   | 92  |
| <a href="#">Xapian::MultiValueSorter</a> ( <a href="#">Sorter</a> subclass which sorts by a several values ) .   | 94  |
| <a href="#">Xapian::NumberValueRangeProcessor</a> (Handle a number range ) . . . . .   | 96  |
| <a href="#">Xapian::PositionIterator</a> (An iterator pointing to items in a list of positions ) .   | 99  |
| <a href="#">Xapian::PostingIterator</a> (An iterator pointing to items in a list of postings ) .   | 101 |

|  |     |
|--|-----|
| <a href="#">Xapian::Query</a> (Class representing a query ) . . . . .  | 105 |
| <a href="#">Xapian::QueryParser</a> (Build a <a href="#">Xapian::Query</a> object from a user query string ) . . . . .                                   | 115 |
| <a href="#">Xapian::RSet</a> (A relevance set (R-Set) ) . . . . .  | 122 |
| <a href="#">Xapian::SimpleStopper</a> (Simple implementation of <a href="#">Stopper</a> class - this will suit most users ) . . . . .                    | 125 |
| <a href="#">Xapian::Sorter</a> (Virtual base class for sorter functor ) . . . . .  | 127 |
| <a href="#">Xapian::Stem</a> (Class representing a stemming algorithm ) . . . . .  | 128 |
| <a href="#">Xapian::Stopper</a> (Base class for stop-word decision functor ) . . . . .   | 131 |
| <a href="#">Xapian::StringAndFrequency</a> (A string with a corresponding frequency ) . . . . .  | 133 |
| <a href="#">Xapian::StringListSerialiser</a> (Class to serialise a list of strings in a form suitable for <a href="#">ValueCountMatchSpy</a> ) . . . . . | 134 |
| <a href="#">Xapian::StringListUnserialiser</a> (Class to unserialise a list of strings serialised by a <a href="#">StringListSerialiser</a> ) . . . . .  | 136 |
| <a href="#">Xapian::StringValueRangeProcessor</a> (Handle a string range ) . . . . .   | 139 |
| <a href="#">Xapian::TermCountMatchSpy</a> (Class for counting the frequencies of terms in the matching documents ) . . . . .                             | 141 |
| <a href="#">Xapian::TermGenerator</a> (Parses a piece of text and generate terms ) . . . . .   | 145 |
| <a href="#">Xapian::TermIterator</a> (An iterator pointing to items in a list of terms ) . . . . .   | 150 |
| <a href="#">Xapian::TradWeight</a> (Traditional probabilistic weighting scheme ) . . . . .   | 154 |
| <a href="#">Xapian::Utf8Iterator</a> (An iterator which returns unicode character values from a UTF-8 encoded string ) . . . . .                         | 158 |
| <a href="#">Xapian::ValueCountMatchSpy</a> (Class for counting the frequencies of values in the matching documents ) . . . . .                           | 163 |
| <a href="#">Xapian::ValueIterator</a> (An iterator pointing to values associated with a document ) . . . . .   | 167 |
| <a href="#">Xapian::ValueRangeProcessor</a> (Base class for value range processors ) . . . . .   | 170 |
| <a href="#">Xapian::Weight</a> (Abstract base class for weighting schemes ) . . . . .  | 172 |
| <a href="#">Xapian::WritableDatabase</a> (This class provides read/write access to a database ) . . . . .  | 176 |

## Chapter 4

# xapian-core File Index

### 4.1 xapian-core File List

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

|  |     |
|--|-----|
| include/xapian.h (Public interfaces for the <a href="#">Xapian</a> library ) . . . . .                           | 187 |
| include/xapian/base.h . . . . .  | ??  |
| include/xapian/database.h (API for working with <a href="#">Xapian</a> databases ) . . . . .                     | 189 |
| include/xapian/dbfactory.h (Factory functions for constructing Database and WritableDatabase objects ) . . . . . | 191 |
| include/xapian/deprecated.h . . . . .  | ??  |
| include/xapian/document.h (API for working with documents ) . . . . .  | 193 |
| include/xapian/enquire.h (API for running queries ) . . . . .  | 194 |
| include/xapian/errorhandler.h (Decide if a Xapian::Error exception should be ignored ) . . . . .                 | 196 |
| include/xapian/expanddecider.h (Allow rejection of terms during ESet generation ) . . . . .                      | 197 |
| include/xapian/matchspy.h (MatchDecider subclasses for use as "match spies" ) . . . . .                          | 198 |
| include/xapian/positioniterator.h (Classes for iterating through position lists )                                | 200 |
| include/xapian/postingiterator.h (Classes for iterating through posting lists )                                  | 201 |
| include/xapian/query.h (Classes for representing a query ) . . . . .   | 202 |
| include/xapian/queryparser.h (Parsing a user query string to build a Xapian::Query object ) . . . . .            | 203 |
| include/xapian/sorter.h (Build sort keys for MSet ordering ) . . . . .   | 205 |
| include/xapian/stem.h (Stemming algorithms ) . . . . .   | 206 |
| include/xapian/termgenerator.h (Parse free text and generate terms ) . . . . .                                   | 207 |
| include/xapian/termiterator.h (Classes for iterating through term lists ) . . . . .                              | 208 |
| include/xapian/types.h (Typedefs for <a href="#">Xapian</a> ) . . . . .  | 209 |
| include/xapian/unicode.h (Unicode and UTF-8 related classes and functions )                                      | 211 |
| include/xapian/valueiterator.h (Classes for iterating through values ) . . . . .                                 | 213 |
| include/xapian/visibility.h . . . . .  | ??  |



## Chapter 5

# xapian-core Page Index

### 5.1 xapian-core Related Pages

Here is a list of all related documentation pages:

Deprecated List . . . . . [215](#)





## Chapter 6

# xapian-core Namespace Documentation

### 6.1 Xapian Namespace Reference

The [Xapian](#) library lives in the [Xapian](#) namespace.

#### Classes

- 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 document in the database - holds data, values, terms, and postings.*
- 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 [MultipleMatchDecider](#)

*Class which applies several match deciders in turn.*

- struct [StringAndFrequency](#)

*A string with a corresponding frequency.*

- class [StringListSerialiser](#)

*Class to serialise a list of strings in a form suitable for [ValueCountMatchSpy](#).*

- class [StringListUnserialiser](#)

*Class to unserialise a list of strings serialised by a [StringListSerialiser](#).*

- class [ValueCountMatchSpy](#)

*Class for counting the frequencies of values in the matching documents.*

- class [TermCountMatchSpy](#)

*Class for counting the frequencies of terms in the matching documents.*

- class [CategorySelectMatchSpy](#)

*MatchSpy for classifying matching documents by their values.*

- class **TermPosWrapper**

- class [PositionIterator](#)

*An iterator pointing to items in a list of positions.*

- class **DocIDWrapper**

- 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 **TermNameWrapper**
- 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 microseconds.*
- 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)
- bool **operator!=** (const [MSetIterator](#) &a, const [MSetIterator](#) &b)
- bool **operator==** (const [ESetIterator](#) &a, const [ESetIterator](#) &b)
- bool **operator!=** (const [ESetIterator](#) &a, const [ESetIterator](#) &b)
- bool **operator==** (const [StringListUnserialiser](#) &a, const [StringListUnserialiser](#) &b)
- bool **operator!=** (const [StringListUnserialiser](#) &a, const [StringListUnserialiser](#) &b)
- 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.*
- XAPIAN\_VISIBILITY\_DEFAULT std::string [sortable\\_serialise](#) (double value)  
*Convert a floating point number to a string, preserving sort order.*
- XAPIAN\_VISIBILITY\_DEFAULT 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)
- bool **operator!=** (const [TermIterator](#) &a, const [TermIterator](#) &b)
- bool **operator==** (const [ValueIterator](#) &a, const [ValueIterator](#) &b)

- bool **operator!=** (const [ValueIterator](#) &a, const [ValueIterator](#) &b)
- XAPIAN\_VISIBILITY\_DEFAULT const char \* [version\\_string](#) ()  
*Report the version string of the library which the program is linked with.*
- XAPIAN\_VISIBILITY\_DEFAULT [XAPIAN\\_DEPRECATED](#) (const char \*xapian\_version\_string())  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*
- XAPIAN\_VISIBILITY\_DEFAULT int [major\\_version](#) ()  
*Report the major version of the library which the program is linked to.*
- XAPIAN\_VISIBILITY\_DEFAULT [XAPIAN\\_DEPRECATED](#) (int xapian\_major\_version())  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*
- XAPIAN\_VISIBILITY\_DEFAULT int [minor\\_version](#) ()  
*Report the minor version of the library which the program is linked to.*
- XAPIAN\_VISIBILITY\_DEFAULT int [revision](#) ()  
*Report the revision of the library which the program is linked to.*

## 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](#) library lives in the [Xapian](#) namespace.

## 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 microseconds.

There are 1 million microseconds in a second, so for example, to set a timeout of 5 seconds use 5000000.

**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 XAPIAN\_VISIBILITY\_DEFAULT int Xapian::major\_version ()**

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

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

**6.1.3.2 XAPIAN\_VISIBILITY\_DEFAULT int Xapian::minor\_version ()**

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



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 PostingIterator & a, const PostingIterator & b) [inline]`

Test inequality of two PostingIterators.

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

Test inequality of two PositionIterators.

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

Test equality of two PostingIterators.

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

Test equality of two PositionIterators.

**6.1.3.7** `XAPIAN_VISIBILITY_DEFAULT int Xapian::revision ()`

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

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

**6.1.3.8** `XAPIAN_VISIBILITY_DEFAULT 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.

#### 6.1.3.9 XAPIAN\_VISIBILITY\_DEFAULT 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.

#### 6.1.3.10 XAPIAN\_VISIBILITY\_DEFAULT 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.

#### 6.1.3.11 XAPIAN\_VISIBILITY\_DEFAULT Xapian::XAPIAN\_DEPRECATED (int xapian\_major\_version())

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

#### Deprecated

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

#### 6.1.3.12 XAPIAN\_VISIBILITY\_DEFAULT Xapian::XAPIAN\_DEPRECATED (const char \* 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.



## Chapter 7

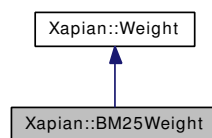
# xapian-core Class Documentation

### 7.1 Xapian::BM25Weight Class Reference

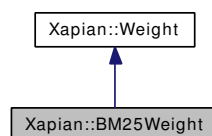
BM25 weighting scheme.

```
#include <enquire.h>
```

Inheritance diagram for Xapian::BM25Weight:



Collaboration 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*

### 7.1.1 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

## 7.1.2 Constructor & Destructor Documentation

### 7.1.2.1 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.

## 7.1.3 Member Function Documentation

### 7.1.3.1 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](#).

### 7.1.3.2 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](#).

### 7.1.3.3 std::string Xapian::BM25Weight::serialise () const [virtual]

Serialise object parameters into a string.

Implements [Xapian::Weight](#).

#### 7.1.3.4 **BM25Weight\* Xapian::BM25Weight::unserialise (const std::string & s) const** [virtual]

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

Implements [Xapian::Weight](#).

#### 7.1.3.5 **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](#).

#### 7.1.3.6 **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](#).

#### 7.1.3.7 **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](#).

#### 7.1.3.8 **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](#).

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

return false if the weight object doesn't need doclength

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

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

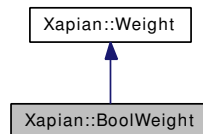
- `include/xapian/enquire.h`

## 7.2 Xapian::BoolWeight Class Reference

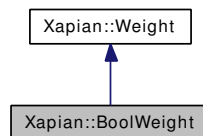
Boolean weighting scheme (everything gets 0).

```
#include <enquire.h>
```

Inheritance diagram for Xapian::BoolWeight:



Collaboration 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*

### 7.2.1 Detailed Description

Boolean weighting scheme (everything gets 0).

### 7.2.2 Member Function Documentation

#### 7.2.2.1 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](#).

#### 7.2.2.2 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](#).

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

Serialise object parameters into a string.

Implements [Xapian::Weight](#).

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

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

Implements [Xapian::Weight](#).

#### 7.2.2.5 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](#).

**7.2.2.6 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](#).

**7.2.2.7 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](#).

**7.2.2.8 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](#).

**7.2.2.9 bool Xapian::BoolWeight::get\_sumpart\_needs\_doclength () const**  
[virtual]

return false if the weight object doesn't need doclength

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

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

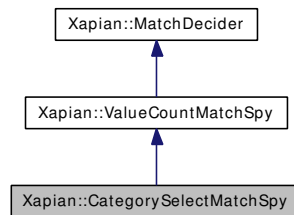
- `include/xapian/enquire.h`

## 7.3 Xapian::CategorySelectMatchSpy Class Reference

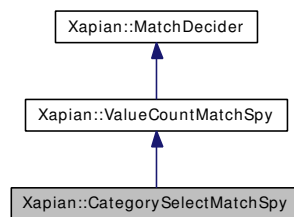
MatchSpy for classifying matching documents by their values.

```
#include <matchspy.h>
```

Inheritance diagram for Xapian::CategorySelectMatchSpy:



Collaboration diagram for Xapian::CategorySelectMatchSpy:



### Public Member Functions

- [CategorySelectMatchSpy](#) ()  
*Default constructor.*
- [CategorySelectMatchSpy](#) (Xapian::valueno valno)  
*Construct a MatchSpy which classifies matching documents based on the values in a particular slot.*
- double [score\\_categorisation](#) (Xapian::valueno valno, double desired\_no\_of\_categories=0.0)  
*Return a score reflecting how "good" a categorisation is.*
- bool [build\\_numeric\\_ranges](#) (Xapian::valueno valno, size\_t max\_ranges)  
*Turn a category containing sort-encoded numeric values into a set of ranges.*

#### 7.3.1 Detailed Description

MatchSpy for classifying matching documents by their values.

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 Xapian::CategorySelectMatchSpy::CategorySelectMatchSpy () [inline]

Default constructor.

#### 7.3.2.2 Xapian::CategorySelectMatchSpy::CategorySelectMatchSpy (Xapian::valueno *valno*) [inline]

Construct a MatchSpy which classifies matching documents based on the values in a particular slot.

Further slots can be added by calling [add\\_slot\(\)](#).

### 7.3.3 Member Function Documentation

#### 7.3.3.1 double Xapian::CategorySelectMatchSpy::score\_categorisation (Xapian::valueno *valno*, double *desired\_no\_of\_categories* = 0.0)

Return a score reflecting how "good" a categorisation is.

If you don't want to show a poor categorisation, or have multiple categories and only space in your user interface to show a few, you want to be able to decide how "good" a categorisation is. We define a good categorisation as one which offers a fairly even split, and (optionally) about a specified number of options.

##### Parameters:

*valno* Value number to look at the categorisation for.

*desired\_no\_of\_categories* The desired number of categories - this is a floating point value, so you can ask for 5.5 if you'd like "about 5 or 6 categories". The default is to desire the number of categories that there actually are, so the score then only reflects how even the split is.

##### Returns:

A score for the categorisation for value *valno* - lower is better, with a perfectly even split across the right number of categories scoring 0.

#### 7.3.3.2 bool Xapian::CategorySelectMatchSpy::build\_numeric\_ranges (Xapian::valueno *valno*, size\_t *max\_ranges*)

Turn a category containing sort-encoded numeric values into a set of ranges.

For "continuous" values (such as price, height, weight, etc), there will usually be too many different values to offer the user, and the user won't want to restrict to an exact value anyway.

This method produces a set of ranges for a particular value number. The ranges replace the category data for value *valno* - the keys are either empty (entry for "no value set"),  $\leq 9$  bytes long (a singleton encoded value), or  $> 9$  bytes long (the first 9 bytes are the encoded range start, the rest the encoded range end).

**Parameters:**

*valno* Value number to produce ranges for.

*max\_ranges* Group into at most this many ranges.

**Returns:**

true if ranges could be built; false if not (e.g. all values the same, no values set, or other reasons).

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

- include/xapian/[matchspy.h](#)

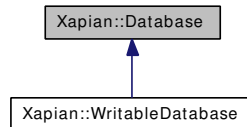


## 7.4 Xapian::Database Class Reference

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

```
#include <database.h>
```

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 &tname) 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 & tname) 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 & tname) const  
*Get the number of documents in the database indexed by a given term.*
- bool [term\\_exists](#) (const std::string & tname) const  
*Check if a given term exists in the database.*
- [Xapian::termcount get\\_collection\\_freq](#) (const std::string & tname) 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 void Xapian::Database::operator= (const Database & *other*)**

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

**7.4.3.3 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.

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

Return a string describing this object.

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

**7.4.3.5 PostingIterator Xapian::Database::postlist\_begin (const std::string & *name*) 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.

**7.4.3.6 PostingIterator Xapian::Database::postlist\_end (const std::string & *name*) const**  
[inline]

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

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

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

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

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

**7.4.3.9 bool Xapian::Database::has\_positions () const**

Does this database have any positional information?

**7.4.3.10 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.

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

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

**7.4.3.12 TermIterator Xapian::Database::allterms\_begin () const**

An iterator which runs across all terms in the database.

**7.4.3.13 TermIterator Xapian::Database::allterms\_end () const** [inline]

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

**7.4.3.14 TermIterator Xapian::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 network databases.

**Parameters:**

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

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

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

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

Get the number of documents in the database.

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

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

**7.4.3.18 Xapian::doclength Xapian::Database::get\_avlength () const**

Get the average length of the documents in the database.

**7.4.3.19 Xapian::doccount Xapian::Database::get\_termfreq (const std::string & *tname*) const**

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

**7.4.3.20 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.

**7.4.3.21 Xapian::termcount Xapian::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.

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

Get the length of a document.

**7.4.3.23 void Xapian::Database::keep\_alive ()**

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

#### 7.4.3.24 **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 for which to retrieve the data.

##### **Returns:**

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

##### **Exceptions:**

*Xapian::DocNotFoundError* The document specified could not be found in the database.

#### 7.4.3.25 **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).

#### 7.4.3.26 **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::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.

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

Corresponding end iterator to synonyms\_begin(term).

**7.4.3.30 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.

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

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

**7.4.3.32 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.

*Xapian::UnimplementedError* will be thrown if the database backend in use doesn't support user-specified metadata.

**7.4.3.33 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.

**Parameters:**

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

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

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

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

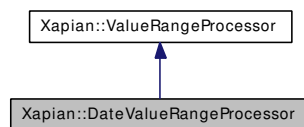
- [include/xapian/database.h](#)

## 7.5 Xapian::DateValueRangeProcessor Class Reference

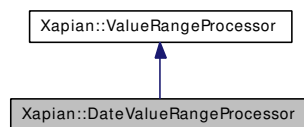
Handle a date range.

```
#include <queryparser.h>
```

Inheritance diagram for Xapian::DateValueRangeProcessor:



Collaboration 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)

*See if <begin>.*

### 7.5.1 Detailed Description

Handle a date range.

Begin and end must be dates in a recognised format.

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 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).

### 7.5.3 Member Function Documentation

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

See if <begin>.

.<end> is a valid date value range.

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:

- include/xapian/[queryparser.h](#)

## 7.6 Xapian::Document Class Reference

A document in the database - holds data, values, terms, and postings.

```
#include <document.h>
```

### 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 [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.*

### 7.6.1 Detailed Description

A document in the database - holds data, values, terms, and postings.

### 7.6.2 Constructor & Destructor Documentation

#### 7.6.2.1 Xapian::Document::Document (const Document & other)

Copying is allowed.

The internals are reference counted, so copying is cheap.

### 7.6.2.2 Xapian::Document::Document ()

Make a new empty [Document](#).

### 7.6.2.3 Xapian::Document::~~Document ()

Destructor.

## 7.6.3 Member Function Documentation

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

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

### 7.6.3.2 std::string Xapian::Document::get\_value (Xapian::value\_no *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.

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

Add a new value.

It will replace any existing value with the same number.

### 7.6.3.4 void Xapian::Document::remove\_value (Xapian::value\_no *valueno*)

Remove any value with the given number.

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

Remove all values associated with the document.

### 7.6.3.6 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.

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

Set data stored in the document.

#### **7.6.3.8 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.

#### **7.6.3.9 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.

#### **7.6.3.10 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

**7.6.3.11 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

**7.6.3.12 void Xapian::Document::clear\_terms ()**

Remove all terms (and postings) from the document.

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

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

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

Iterator for the terms in this document.

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

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

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

Count the values in this document.

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

Iterator for the values in this document.

**7.6.3.18 ValueIterator Xapian::Document::values\_end () const**

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

**7.6.3.19 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.

**7.6.3.20 std::string Xapian::Document::get\_description () const**

Return a string describing this object.

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

- include/xapian/[document.h](#)

## 7.7 Xapian::Enquire Class Reference

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

```
#include <enquire.h>
```

### 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 ascending=true)  
*Set the sorting to be by value only.*
- void `set_sort_by_key` (`Xapian::Sorter` \*sorter, bool ascending=true)  
*Set the sorting to be by key generated from values only.*
- void `set_sort_by_value_then_relevance` (`Xapian::valueno` sort\_key, bool ascending=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 ascending=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 ascending=true)  
*Set the sorting to be by relevance then value.*
- void `set_sort_by_relevance_then_key` (`Xapian::Sorter` \*sorter, bool ascending=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
- `XAPIAN_DEPRECATED` (static const int include\_query\_terms)  
*Deprecated in Xapian 1.0.0, use INCLUDE\_QUERY\_TERMS instead.*
- `XAPIAN_DEPRECATED` (static const int use\_exact\_termfreq)  
*Deprecated in Xapian 1.0.0, use USE\_EXACT\_TERM\_FREQ instead.*
- `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](#)().*
- [XAPIAN\\_DEPRECATED](#) (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\\_TERMREQ](#) = 2

### 7.7.1 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.

## 7.7.2 Constructor & Destructor Documentation

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

Copying is allowed (and is cheap).

### 7.7.2.2 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.

### 7.7.2.3 Xapian::Enquire::~~Enquire ()

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

## 7.7.3 Member Function Documentation

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

Assignment is allowed (and is cheap).

### 7.7.3.2 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.

**7.7.3.3 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.

**7.7.3.4 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.

**7.7.3.5 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).

### 7.7.3.6 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".

### 7.7.3.7 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).

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

Set the sorting to be by relevance only.

This is the default.



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

Set the sorting to be by value only.

NB sorting of values uses a string comparison, so you'll need to store numbers padded with leading zeros or spaces, or with the number of digits prepended.

#### Parameters:

*sort\_key* value number to sort on.

*ascending* If true, documents values which sort higher by string compare are better. If false, the sort order is reversed. (default true)

### 7.7.3.10 void Xapian::Enquire::set\_sort\_by\_key (Xapian::Sorter \* *sorter*, bool *ascending* = true)

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

#### Parameters:

*sorter* The functor to use for generating keys.

*ascending* If true, documents values which sort higher by string compare are better. If false, the sort order is reversed. (default true)

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

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

NB sorting of values uses a string comparison, so you'll need to store numbers padded with leading zeros or spaces, or with the number of digits prepended.

#### Parameters:

*sort\_key* value number to sort on.

*ascending* If true, documents values which sort higher by string compare are better. If false, the sort order is reversed. (default true)

### 7.7.3.12 void Xapian::Enquire::set\_sort\_by\_key\_then\_relevance (Xapian::Sorter \* *sorter*, bool *ascending* = 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.

*ascending* If true, keys which sort higher by string compare are better. If false, the sort order is reversed. (default true)

#### 7.7.3.13 void Xapian::Enquire::set\_sort\_by\_relevance\_then\_value (Xapian::value no\_sort\_key, bool ascending = true)

Set the sorting to be by relevance then value.

NB sorting of values uses a string comparison, so you'll need to 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:

*no\_sort\_key* value number to sort on.

*ascending* If true, documents values which sort higher by string compare are better. If false, the sort order is reversed. (default true)

#### 7.7.3.14 void Xapian::Enquire::set\_sort\_by\_relevance\_then\_key (Xapian::Sorter \* sorter, bool ascending = 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.

*ascending* If true, keys which sort higher by string compare are better. If false, the sort order is reversed. (default true)

#### 7.7.3.15 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.

*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.

#### 7.7.3.16 Xapian::Enquire::XAPIAN\_DEPRECATED (static const int include\_query\_terms)

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

#### 7.7.3.17 Xapian::Enquire::XAPIAN\_DEPRECATED (static const int use\_exact\_termfreq)

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

#### 7.7.3.18 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.

**7.7.3.19** `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.

**7.7.3.20** `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.

**7.7.3.21 TermIterator Xapian::Enquire::get\_matching\_terms\_end (Xapian::docid) const [inline]**

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

**7.7.3.22 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.

**7.7.3.23** `TermIterator Xapian::Enquire::get_matching_terms_end (const MSetIterator &) const` `[inline]`

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

**7.7.3.24** `Xapian::Enquire::XAPIAN_DEPRECATED (void 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.

**7.7.3.25** `std::string Xapian::Enquire::get_description () const`

Return a string describing this object.

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

- `include/xapian/enquire.h`

## 7.8 Xapian::ErrorHandler Class Reference

Decide if a Xapian::Error exception should be ignored.

```
#include <errorhandler.h>
```

### 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.*

### 7.8.1 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).

### 7.8.2 Constructor & Destructor Documentation

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

Default constructor.

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

We require a virtual destructor because we have virtual methods.

### 7.8.3 Member Function Documentation

#### 7.8.3.1 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:

- `include/xapian/errorhandler.h`



## 7.9 Xapian::ESet Class Reference

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

```
#include <enquire.h>
```

### 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`

### 7.9.1 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\(\)](#).

### 7.9.2 Constructor & Destructor Documentation

#### 7.9.2.1 `Xapian::ESet::ESet ()`

Construct an empty [ESet](#).

#### 7.9.2.2 `Xapian::ESet::~~ESet ()`

Destructor.

#### 7.9.2.3 `Xapian::ESet::ESet (const ESet & other)`

Copying is allowed (and is cheap).

### 7.9.3 Member Function Documentation

#### 7.9.3.1 `void Xapian::ESet::operator= (const ESet & other)`

Assignment is allowed (and is cheap).

#### 7.9.3.2 `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\(\)](#)

#### 7.9.3.3 `Xapian::termcount Xapian::ESet::size () const`

The number of terms in this E-Set.

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

Required to allow use as an STL container.

**7.9.3.5 bool Xapian::ESet::empty () const**

Test if this E-Set is empty.

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

Swap the E-Set we point to with another.

**7.9.3.7 ESetIterator Xapian::ESet::begin () const**

Iterator for the terms in this E-Set.

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

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

**7.9.3.9 ESetIterator Xapian::ESet::back () const**

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

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

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

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

Return a string describing this object.

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

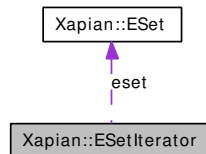
- [include/xapian/enquire.h](#)

## 7.10 Xapian::ESetIterator Class Reference

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

```
#include <enquire.h>
```

Collaboration diagram for Xapian::ESetIterator:



### Public Types

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

### 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

- `class ESet`
- `bool operator== (const ESetIterator &a, const ESetIterator &b)`
- `bool operator!= (const ESetIterator &a, const ESetIterator &b)`

### 7.10.1 Detailed Description

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

### 7.10.2 Member Typedef Documentation

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

Allow use as an STL iterator.

### 7.10.3 Constructor & Destructor Documentation

#### 7.10.3.1 `Xapian::ESetIterator::ESetIterator ()` `[inline]`

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

#### 7.10.3.2 `Xapian::ESetIterator::ESetIterator (const ESetIterator & other)` `[inline]`

Copying is allowed (and is cheap).

### 7.10.4 Member Function Documentation

#### 7.10.4.1 `void Xapian::ESetIterator::operator= (const ESetIterator & other)` `[inline]`

Assignment is allowed (and is cheap).

**7.10.4.2 ESetIterator& Xapian::ESetIterator::operator++ () [inline]**

Advance the iterator.

**7.10.4.3 ESetIterator Xapian::ESetIterator::operator++ (int) [inline]**

Advance the iterator (postfix variant).

**7.10.4.4 ESetIterator& Xapian::ESetIterator::operator-- () [inline]**

Decrement the iterator.

**7.10.4.5 ESetIterator Xapian::ESetIterator::operator-- (int) [inline]**

Decrement the iterator (postfix variant).

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

Get the term for the current position.

**7.10.4.7 Xapian::weight Xapian::ESetIterator::get\_weight () const**

Get the weight of the term at the current position.

**7.10.4.8 std::string Xapian::ESetIterator::get\_description () const**

Return a string describing this object.

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

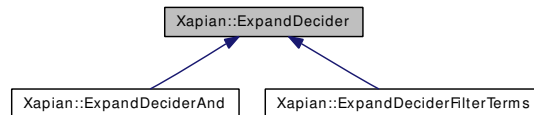
- [include/xapian/enquire.h](#)

## 7.11 Xapian::ExpandDecider Class Reference

Virtual base class for expand decider functor.

```
#include <expanddecider.h>
```

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.*

#### 7.11.1 Detailed Description

Virtual base class for expand decider functor.

#### 7.11.2 Constructor & Destructor Documentation

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

Virtual destructor, because we have virtual methods.

#### 7.11.3 Member Function Documentation

##### 7.11.3.1 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:

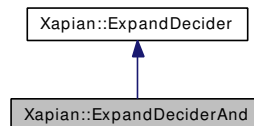
- include/xapian/[expanddecider.h](#)

## 7.12 Xapian::ExpandDeciderAnd Class Reference

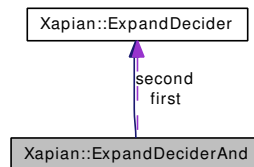
[ExpandDecider](#) subclass which rejects terms using two [ExpandDeciders](#).

```
#include <expanddecider.h>
```

Inheritance diagram for Xapian::ExpandDeciderAnd:



Collaboration 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](#)?*

### 7.12.1 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.



## 7.12.2 Constructor & Destructor Documentation

**7.12.2.1 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider & *first\_*, const ExpandDecider & *second\_*)** [inline]

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

**7.12.2.2 Xapian::ExpandDeciderAnd::ExpandDeciderAnd (const ExpandDecider \* *first\_*, const ExpandDecider \* *second\_*)** [inline]

Compatibility method.

## 7.12.3 Member Function Documentation

**7.12.3.1 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:

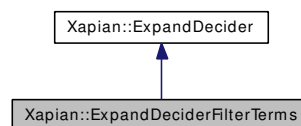
- [include/xapian/expanddecider.h](#)

## 7.13 Xapian::ExpandDeciderFilterTerms Class Reference

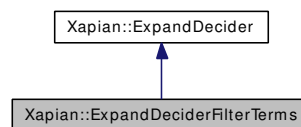
[ExpandDecider](#) subclass which rejects terms in a specified list.

```
#include <expanddecider.h>
```

Inheritance diagram for Xapian::ExpandDeciderFilterTerms:



Collaboration 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](#)?*

#### 7.13.1 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](#).

#### 7.13.2 Constructor & Destructor Documentation

- 7.13.2.1 `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 **`).

### 7.13.3 Member Function Documentation

#### 7.13.3.1 `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:

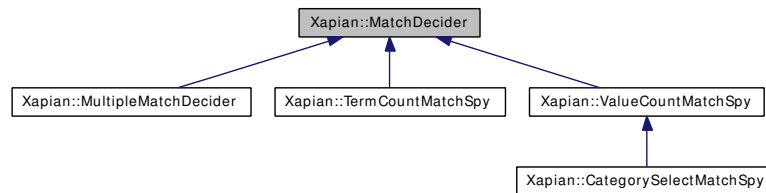
- `include/xapian/expanddecider.h`

## 7.14 Xapian::MatchDecider Class Reference

Base class for matcher decision functor.

```
#include <enquire.h>
```

Inheritance diagram for Xapian::MatchDecider:



### 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.*

#### 7.14.1 Detailed Description

Base class for matcher decision functor.

#### 7.14.2 Constructor & Destructor Documentation

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

Destructor.

#### 7.14.3 Member Function Documentation

##### 7.14.3.1 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](#).

Implemented in [Xapian::MultipleMatchDecider](#), [Xapian::ValueCountMatchSpy](#), and [Xapian::TermCountMatchSpy](#).

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

- `include/xapian/enquire.h`

## 7.15 Xapian::MSet Class Reference

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

```
#include <enquire.h>
```

### Public Types

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

### 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**

### 7.15.1 Detailed Description

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

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

### 7.15.2 Member Typedef Documentation

#### 7.15.2.1 `typedef MSetIterator Xapian::MSet::value_type`

Allow use as an STL container.

### 7.15.3 Constructor & Destructor Documentation

#### 7.15.3.1 `Xapian::MSet::MSet ()`

Create an empty [Xapian::MSet](#).

#### 7.15.3.2 `Xapian::MSet::~~MSet ()`

Destroy a [Xapian::MSet](#).

#### 7.15.3.3 `Xapian::MSet::MSet (const MSet & other)`

Copying is allowed (and is cheap).

### 7.15.4 Member Function Documentation

#### 7.15.4.1 `void Xapian::MSet::operator= (const MSet & other)`

Assignment is allowed (and is cheap).



#### 7.15.4.2 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.

#### 7.15.4.3 void Xapian::MSet::fetch (const MSetIterator & *item*) const

Fetch the single item specified.

#### 7.15.4.4 void Xapian::MSet::fetch () const

Fetch all the items in the [MSet](#).

#### 7.15.4.5 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.

#### 7.15.4.6 Xapian::percent Xapian::MSet::convert\_to\_percent (const MSetIterator & *it*) const

Return the percentage score for a particular item.

#### 7.15.4.7 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.

**7.15.4.8 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.

**7.15.4.9 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](#).

**7.15.4.10 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.

**7.15.4.11 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.

**7.15.4.12 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.

#### 7.15.4.13 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.

#### 7.15.4.14 Xapian::weight Xapian::MSet::get\_max\_attained () const

The greatest weight which is attained by any document in the database.

If `firstitem == 0`, this is the weight of the first entry in items.

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.

#### 7.15.4.15 Xapian::doccount Xapian::MSet::size () const

The number of items in this [MSet](#).

#### 7.15.4.16 Xapian::doccount Xapian::MSet::max\_size () const `[inline]`

Required to allow use as an STL container.

#### 7.15.4.17 bool Xapian::MSet::empty () const

Test if this [MSet](#) is empty.

#### 7.15.4.18 void Xapian::MSet::swap (MSet & *other*)

Swap the [MSet](#) we point to with another.

#### 7.15.4.19 MSetIterator Xapian::MSet::begin () const

Iterator for the terms in this [MSet](#).

#### 7.15.4.20 MSetIterator Xapian::MSet::end () const

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

**7.15.4.21 MSetIterator Xapian::MSet::back () const**

Iterator pointing to the last element of this [MSet](#).

**7.15.4.22 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.

**7.15.4.23 std::string Xapian::MSet::get\_description () const**

Return a string describing this object.

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

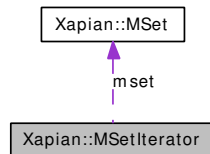
- [include/xapian/enquire.h](#)

## 7.16 Xapian::MSetIterator Class Reference

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

```
#include <enquire.h>
```

Collaboration diagram for Xapian::MSetIterator:



### Public Types

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

### 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

- class **MSet**
- bool **operator==** (const [MSetIterator](#) &a, const [MSetIterator](#) &b)
- bool **operator!=** (const [MSetIterator](#) &a, const [MSetIterator](#) &b)

### 7.16.1 Detailed Description

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

This is used for access to individual results of a match.

### 7.16.2 Member Typedef Documentation

#### 7.16.2.1 `typedef std::bidirectional_iterator_tag` `Xapian::MSetIterator::iterator_category`

Allow use as an STL iterator.

### 7.16.3 Constructor & Destructor Documentation

#### 7.16.3.1 Xapian::MSetIterator::MSetIterator () [inline]

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

#### 7.16.3.2 Xapian::MSetIterator::MSetIterator (const MSetIterator & *other*) [inline]

Copying is allowed (and is cheap).

### 7.16.4 Member Function Documentation

#### 7.16.4.1 void Xapian::MSetIterator::operator= (const MSetIterator & *other*) [inline]

Assignment is allowed (and is cheap).

#### 7.16.4.2 MSetIterator& Xapian::MSetIterator::operator++ () [inline]

Advance the iterator.

#### 7.16.4.3 MSetIterator Xapian::MSetIterator::operator++ (int) [inline]

Advance the iterator (postfix variant).

#### 7.16.4.4 MSetIterator& Xapian::MSetIterator::operator-- () [inline]

Decrement the iterator.

#### 7.16.4.5 MSetIterator Xapian::MSetIterator::operator-- (int) [inline]

Decrement the iterator (postfix variant).

#### 7.16.4.6 Xapian::docid Xapian::MSetIterator::operator \* () const

Get the document ID for the current position.

#### 7.16.4.7 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.

**7.16.4.8 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.

**7.16.4.9 Xapian::weight Xapian::MSetIterator::get\_weight () const**

Get the weight of the document at the current position.

**7.16.4.10 std::string Xapian::MSetIterator::get\_collapse\_key () const**

Get the collapse key for this document.

**7.16.4.11 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).

**7.16.4.12 Xapian::percent Xapian::MSetIterator::get\_percent () const**

This returns the weight of the document as a percentage score.



The return value will be in the range 0 to 100: 0 meaning that the item did not match the query at all.

#### **7.16.4.13   `std::string Xapian::MSetIterator::get_description () const`**

Return a string describing this object.

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

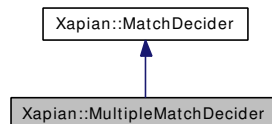
- `include/xapian/enquire.h`

## 7.17 Xapian::MultipleMatchDecider Class Reference

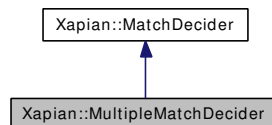
Class which applies several match deciders in turn.

```
#include <matchspy.h>
```

Inheritance diagram for Xapian::MultipleMatchDecider:



Collaboration diagram for Xapian::MultipleMatchDecider:



### Public Member Functions

- void **append** (const **MatchDecider** \*decider)  
*Add a match decider to the end of the list to be called.*
- bool **operator()** (const **Xapian::Document** &doc) const  
*Implementation of virtual operator().*

### 7.17.1 Detailed Description

Class which applies several match deciders in turn.

### 7.17.2 Member Function Documentation

#### 7.17.2.1 void Xapian::MultipleMatchDecider::append (const MatchDecider \*decider) [inline]

Add a match decider to the end of the list to be called.

Note that the caller must ensure that the decider is not deleted before it is used - the **MultipleMatchDecider** keeps a pointer to the supplied decider.

### 7.17.2.2 `bool Xapian::MultipleMatchDecider::operator() (const Xapian::Document & doc) const` [virtual]

Implementation of virtual operator().

This implementation calls the deciders in turn, until one of them returns false, or all have been called. It returns true iff all the deciders return true.

Implements [Xapian::MatchDecider](#).

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

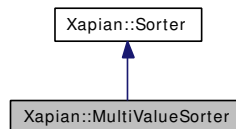
- `include/xapian/matchspy.h`

## 7.18 Xapian::MultiValueSorter Class Reference

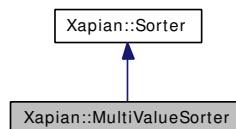
[Sorter](#) subclass which sorts by a several values.

```
#include <sorter.h>
```

Inheritance diagram for Xapian::MultiValueSorter:



Collaboration 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)`

#### 7.18.1 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.

#### 7.18.2 Member Function Documentation

**7.18.2.1** `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:

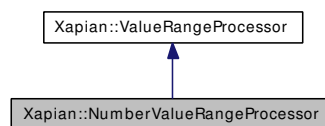
- `include/xapian/sorter.h`

## 7.19 Xapian::NumberValueRangeProcessor Class Reference

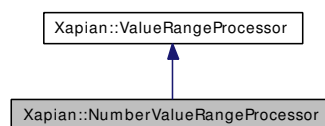
Handle a number range.

```
#include <queryparser.h>
```

Inheritance diagram for Xapian::NumberValueRangeProcessor:



Collaboration 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)  
*See if <begin>.*

#### 7.19.1 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).

## 7.19.2 Constructor & Destructor Documentation

### 7.19.2.1 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno *valno\_*) [inline]

Constructor.

#### Parameters:

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

### 7.19.2.2 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.

## 7.19.3 Member Function Documentation

### 7.19.3.1 Xapian::valueno Xapian::NumberValueRangeProcessor::operator() (std::string & *begin*, std::string & *end*) [virtual]

See if <begin>.

.<end> is a valid numeric value range.

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:

- [include/xapian/queryparser.h](#)



## 7.20 Xapian::PositionIterator Class Reference

An iterator pointing to items in a list of positions.

```
#include <positioniterator.h>
```

### 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)
- std::string [get\\_description](#) () const  
*Return a string describing this object.*

### Friends

- class **PostingIterator**
- class **TermIterator**
- class **Database**
- bool [operator==](#) (const [PositionIterator](#) &a, const [PositionIterator](#) &b)  
*Test equality of two PositionIterators.*

### 7.20.1 Detailed Description

An iterator pointing to items in a list of positions.

### 7.20.2 Constructor & Destructor Documentation

#### 7.20.2.1 Xapian::PositionIterator::PositionIterator ()

Default constructor - for declaring an uninitialised iterator.

#### 7.20.2.2 Xapian::PositionIterator::~~PositionIterator ()

Destructor.

#### 7.20.2.3 Xapian::PositionIterator::PositionIterator (const PositionIterator & o)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

### 7.20.3 Member Function Documentation

#### 7.20.3.1 void Xapian::PositionIterator::operator= (const PositionIterator & o)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

#### 7.20.3.2 std::string Xapian::PositionIterator::get\_description () const

Return a string describing this object.

### 7.20.4 Friends And Related Function Documentation

#### 7.20.4.1 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:

- include/xapian/[positioniterator.h](#)

## 7.21 Xapian::PostingIterator Class Reference

An iterator pointing to items in a list of postings.

```
#include <postingiterator.h>
```

### Public Types

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

### 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)  
*Skip the iterator to document did, or the first document after did if did isn't in the list of documents being iterated.*
- [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

- class **Database**
- bool [operator==](#) (const [PostingIterator](#) &a, const [PostingIterator](#) &b)

Test equality of two *PostingIterators*.

### 7.21.1 Detailed Description

An iterator pointing to items in a list of postings.

### 7.21.2 Member Typedef Documentation

#### 7.21.2.1 typedef std::input\_iterator\_tag Xapian::PostingIterator::iterator\_category

Allow use as an STL iterator.

### 7.21.3 Constructor & Destructor Documentation

#### 7.21.3.1 Xapian::PostingIterator::PostingIterator ()

Default constructor - for declaring an uninitialised iterator.

#### 7.21.3.2 Xapian::PostingIterator::~~PostingIterator ()

Destructor.

#### 7.21.3.3 Xapian::PostingIterator::PostingIterator (const PostingIterator &other)

Copying is allowed.

The internals are reference counted, so copying is also cheap.

## 7.21.4 Member Function Documentation

### 7.21.4.1 void Xapian::PostingIterator::operator= (const PostingIterator & *other*)

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

### 7.21.4.2 void Xapian::PostingIterator::skip\_to (Xapian::docid *did*)

Skip the iterator to document *did*, or the first document after *did* if *did* isn't in the list of documents being iterated.

### 7.21.4.3 Xapian::docid Xapian::PostingIterator::operator \* () const

Get the document id at the current position in the postlist.

### 7.21.4.4 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.

### 7.21.4.5 Xapian::termcount Xapian::PostingIterator::get\_wdf () const

Get the within document frequency of the document at the current position in the postlist.

### 7.21.4.6 PositionIterator Xapian::PostingIterator::positionlist\_begin () const

Return [PositionIterator](#) pointing to start of positionlist for current document.

### 7.21.4.7 PositionIterator Xapian::PostingIterator::positionlist\_end () const [inline]

Return [PositionIterator](#) pointing to end of positionlist for current document.

### 7.21.4.8 std::string Xapian::PostingIterator::get\_description () const

Return a string describing this object.

## 7.21.5 Friends And Related Function Documentation

### 7.21.5.1 `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:

- include/xapian/[postingiterator.h](#)

## 7.22 Xapian::Query Class Reference

Class representing a query.

```
#include <query.h>
```

Collaboration diagram for Xapian::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.*
- typedef `std::vector< Internal * >` `subquery_list`  
*The container type for storing pointers to subqueries.*
- typedef `int` `op_t`  
*Type storing the operation.*

### Public Member Functions

- `Query` (const `Query` &copyyme)  
*Copy constructor.*
- `Query` & `operator=` (const `Query` &copyyme)  
*Assignment.*
- `Query` ()  
*Default constructor: makes an empty query which matches no documents.*
- `~Query` ()  
*Destructor.*
- `Query` (const `std::string` &tname\_, `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.*
- `XAPIAN_DEPRECATED (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.*
- `Internal (const Query::Internal &copyme)`  
*Copy constructor.*
- `void operator= (const Query::Internal &copyme)`  
*Assignment.*



- [Internal](#) (const std::string &tname\_, [Xapian::termcount](#) wqf\_=1, [Xapian::termpos](#) term\_pos\_=0)  
*A query consisting of a single term.*
- [Internal](#) (op\_t op\_, [Xapian::termcount](#) parameter)  
*Create internals given only the operator and a parameter.*
- [Internal](#) (op\_t op\_, [Xapian::valueno](#) valno, const std::string &begin, const std::string &end)  
*Construct a range query on a document value.*
- [Internal](#) (op\_t op\_, [Xapian::valueno](#) valno, const std::string &value)  
*Construct a value greater-than-or-equal query on a document value.*
- [~Internal](#) ()  
*Destructor.*
- void [add\\_subquery](#) (const Query::Internal \*subq)  
*Add a subquery.*
- void [set\\_dbl\\_parameter](#) (double dbl\_parameter\_)
- double [get\\_dbl\\_parameter](#) () const
- Query::Internal \* [end\\_construction](#) ()  
*Finish off the construction.*
- std::string [serialise](#) () const  
*Return a string in an easily parsed form which contains all the information in a query.*
- std::string [get\\_description](#) () const  
*Return a string describing this object.*
- [Xapian::termcount](#) [get\\_parameter](#) () const  
*Get the numeric parameter used in this query.*
- [Xapian::termcount](#) [get\\_length](#) () const  
*Get the length of the query, used by some ranking formulae.*
- [TermIterator](#) [get\\_terms](#) () const  
*Return an iterator over all the terms in the query, in order of termpos.*

## Static Public Member Functions

- static Xapian::Query::Internal \* [unserialise](#) (const std::string &s)

## 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.*
- static const int **OP\_LEAF** = -1

## Friends

- class **::LocalSubMatch**
- class **::MultiMatch**
- class **::QueryOptimiser**
- struct **::SortPosName**
- class **Query**

### 7.22.1 Detailed Description

Class representing a query.

Queries are represented as a tree of objects.

### 7.22.2 Member Typedef Documentation

#### 7.22.2.1 `typedef std::vector<Internal*> Xapian::Query::subquery_list`

The container type for storing pointers to subqueries.

#### 7.22.2.2 `typedef int Xapian::Query::op_t`

Type storing the operation.

### 7.22.3 Member Enumeration Documentation

#### 7.22.3.1 `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.

## 7.22.4 Constructor & Destructor Documentation

### 7.22.4.1 Xapian::Query::Query (const Query & *copyme*)

Copy constructor.

### 7.22.4.2 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.

### 7.22.4.3 Xapian::Query::~~Query ()

Destructor.

#### 7.22.4.4 **Xapian::Query::Query** (const std::string & *tname\_*, Xapian::termcount *wqf\_* = 1, Xapian::termpos *pos\_* = 0)

A query consisting of a single term.

#### 7.22.4.5 **Xapian::Query::Query** (Query::op *op\_*, const Query & *left*, const Query & *right*)

A query consisting of two subqueries, opp-ed together.

#### 7.22.4.6 **Xapian::Query::Query** (Query::op *op\_*, const std::string & *left*, const std::string & *right*)

A query consisting of two termnames opp-ed together.

#### 7.22.4.7 **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.

#### 7.22.4.8 **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.

#### 7.22.4.9 **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.

#### 7.22.4.10 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.

#### 7.22.4.11 Xapian::Query::~~Internal ()

Destructor.

## 7.22.5 Member Function Documentation

### 7.22.5.1 Query& Xapian::Query::operator= (const Query & copyme)

Assignment.

### 7.22.5.2 Xapian::Query::XAPIAN\_DEPRECATED (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 with a parameter value.

### 7.22.5.3 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\(\)](#).

**7.22.5.4 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.

**7.22.5.5 TermIterator Xapian::Query::get\_terms\_end () const [inline]**

Return a [Xapian::TermIterator](#) to the end of the list of terms in the query.

**7.22.5.6 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).

**7.22.5.7 std::string Xapian::Query::get\_description () const**

Return a string describing this object.

**7.22.5.8 Xapian::Query::Internal (const Query::Internal & copyme)**

Copy constructor.

**7.22.5.9 void Xapian::Query::operator= (const Query::Internal & copyme)**

Assignment.

**7.22.5.10 Xapian::Query::Internal (const std::string & *tname\_*,  
Xapian::termcount *wqf\_* = 1, Xapian::termpos *term\_pos\_* = 0)  
[explicit]**

A query consisting of a single term.

**7.22.5.11 Xapian::Query::Internal (op\_t *op\_*, Xapian::termcount *parameter*)**

Create internals given only the operator and a parameter.

**7.22.5.12 Xapian::Query::Internal (op\_t *op\_*, Xapian::valueno *valno*, const  
std::string & *begin*, const std::string & *end*)**

Construct a range query on a document value.

**7.22.5.13 Xapian::Query::Internal (op\_t op\_, Xapian::valueo valno, const std::string & value)**

Construct a value greater-than-or-equal query on a document value.

**7.22.5.14 void Xapian::Query::add\_subquery (const Query::Internal \* subq)**

Add a subquery.

**7.22.5.15 Query::Internal\* Xapian::Query::end\_construction ()**

Finish off the construction.

**7.22.5.16 std::string Xapian::Query::serialise () const [inline]**

Return a string in an easily parsed form which contains all the information in a query.

**7.22.5.17 std::string Xapian::Query::get\_description () const**

Return a string describing this object.

**7.22.5.18 Xapian::termcount Xapian::Query::get\_parameter () const [inline]**

Get the numeric parameter used in this query.

This is used by the [QueryParser](#) to get the value number for VALUE\_RANGE queries. It should be replaced by a public method on the [Query](#) class at some point, but the API which should be used for that is unclear, so this is a temporary workaround.

**7.22.5.19 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\(\)](#).

**7.22.5.20 TermIterator Xapian::Query::get\_terms () const**

Return an iterator over 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.

## 7.22.6 Member Data Documentation

### 7.22.6.1 Xapian::Query Xapian::Query::MatchAll [static]

A query which matches all documents in the database.

### 7.22.6.2 Xapian::Query Xapian::Query::MatchNothing [static]

A query which matches no documents.

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

- [include/xapian/query.h](#)



## 7.23 Xapian::QueryParser Class Reference

Build a [Xapian::Query](#) object from a user query string.

```
#include <queryparser.h>
```

### 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](#)  
}
- 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 boolean operator.

- [Query::op get\\_default\\_op](#) () const  
*Get the default boolean 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.*

### 7.23.1 Detailed Description

Build a [Xapian::Query](#) object from a user query string.

### 7.23.2 Member Enumeration Documentation

#### 7.23.2.1 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\*).

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.

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.

### 7.23.3 Constructor & Destructor Documentation

#### 7.23.3.1 Xapian::QueryParser::QueryParser (const QueryParser & o)

Copy constructor.

### 7.23.3.2 Xapian::QueryParser::QueryParser ()

Default constructor.

### 7.23.3.3 Xapian::QueryParser::~~QueryParser ()

Destructor.

## 7.23.4 Member Function Documentation

### 7.23.4.1 QueryParser& Xapian::QueryParser::operator= (const QueryParser & o)

Assignment.

### 7.23.4.2 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.

### 7.23.4.3 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.

**7.23.4.4 void Xapian::QueryParser::set\_stopper (const Stopper \* stop = NULL)**

Set the stopper.

**7.23.4.5 void Xapian::QueryParser::set\_default\_op (Query::op default\_op)**

Set the default boolean operator.

**7.23.4.6 Query::op Xapian::QueryParser::get\_default\_op () const**

Get the default boolean operator.

**7.23.4.7 void Xapian::QueryParser::set\_database (const Database & db)**

Specify the database being searched.

**7.23.4.8 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 [QueryParser](#) should support. Combine multiple values with bitwise-or (`|`).

*default\_prefix* The default term prefix to use (default none). For example, you can pass "A" when parsing an "Author" field.

**7.23.4.9 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 as the default prefix. If you do this and also specify the `default_prefix` parameter to `parse_query()`, then the `default_prefix` parameter will override.

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

#### 7.23.4.10 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

**7.23.4.11 TermIterator Xapian::QueryParser::stoplist\_begin () const**

Iterate over terms omitted from the query as stopwords.

**7.23.4.12 TermIterator Xapian::QueryParser::unstem\_begin (const std::string & term) const**

Iterate over unstemmed forms of the given (stemmed) term used in the query.

**7.23.4.13 void Xapian::QueryParser::add\_valuerangeprocessor (Xapian::ValueRangeProcessor \* vrproc)**

Register a [ValueRangeProcessor](#).

**7.23.4.14 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.

**7.23.4.15 std::string Xapian::QueryParser::get\_description () const**

Return a string describing this object.

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

- [include/xapian/queryparser.h](#)

## 7.24 Xapian::RSet Class Reference

A relevance set (R-Set).

```
#include <enquire.h>
```

### 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**

### 7.24.1 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.

### 7.24.2 Constructor & Destructor Documentation

#### 7.24.2.1 Xapian::RSet::RSet (const RSet & *rset*)

Copy constructor.

#### 7.24.2.2 Xapian::RSet::RSet ()

Default constructor.

#### 7.24.2.3 Xapian::RSet::~~RSet ()

Destructor.

### 7.24.3 Member Function Documentation

#### 7.24.3.1 void Xapian::RSet::operator= (const RSet & *rset*)

Assignment operator.

#### 7.24.3.2 Xapian::doccount Xapian::RSet::size () const

The number of documents in this R-Set.

#### 7.24.3.3 bool Xapian::RSet::empty () const

Test if this R-Set is empty.

#### 7.24.3.4 void Xapian::RSet::add\_document (Xapian::docid *did*)

Add a document to the relevance set.

**7.24.3.5** `void Xapian::RSet::add_document (const Xapian::MSetIterator & i)`  
[inline]

Add a document to the relevance set.

**7.24.3.6** `void Xapian::RSet::remove_document (Xapian::docid did)`

Remove a document from the relevance set.

**7.24.3.7** `void Xapian::RSet::remove_document (const Xapian::MSetIterator & i)` [inline]

Remove a document from the relevance set.

**7.24.3.8** `bool Xapian::RSet::contains (Xapian::docid did) const`

Test if a given document in the relevance set.

**7.24.3.9** `bool Xapian::RSet::contains (const Xapian::MSetIterator & i) const`  
[inline]

Test if a given document in the relevance set.

**7.24.3.10** `std::string Xapian::RSet::get_description () const`

Return a string describing this object.

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

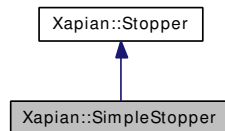
- include/xapian/[enquire.h](#)

## 7.25 Xapian::SimpleStopper Class Reference

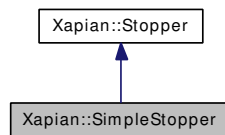
Simple implementation of [Stopper](#) class - this will suit most users.

```
#include <queryparser.h>
```

Inheritance diagram for Xapian::SimpleStopper:



Collaboration 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.*

### 7.25.1 Detailed Description

Simple implementation of [Stopper](#) class - this will suit most users.

## 7.25.2 Constructor & Destructor Documentation

### 7.25.2.1 Xapian::SimpleStopper::SimpleStopper () [inline]

Default constructor.

### 7.25.2.2 template<class Iterator> Xapian::SimpleStopper::SimpleStopper (Iterator *begin*, Iterator *end*) [inline]

Initialise from a pair of iterators.

### 7.25.2.3 virtual Xapian::SimpleStopper::~~SimpleStopper () [inline, virtual]

Destructor.

## 7.25.3 Member Function Documentation

### 7.25.3.1 void Xapian::SimpleStopper::add (const std::string & *word*) [inline]

Add a single stop word.

### 7.25.3.2 virtual bool Xapian::SimpleStopper::operator() (const std::string & *term*) const [inline, virtual]

Is term a stop-word?

Implements [Xapian::Stopper](#).

### 7.25.3.3 virtual std::string Xapian::SimpleStopper::get\_description () const [virtual]

Return a string describing this object.

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

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

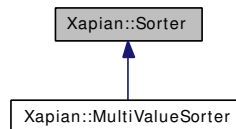
- [include/xapian/queryparser.h](#)

## 7.26 Xapian::Sorter Class Reference

Virtual base class for sorter functor.

```
#include <sorter.h>
```

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.*

### 7.26.1 Detailed Description

Virtual base class for sorter functor.

### 7.26.2 Constructor & Destructor Documentation

#### 7.26.2.1 virtual Xapian::Sorter::~~Sorter () [virtual]

Virtual destructor, because we have virtual methods.

### 7.26.3 Member Function Documentation

#### 7.26.3.1 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:

- include/xapian/[sorter.h](#)

## 7.27 Xapian::Stem Class Reference

Class representing a stemming algorithm.

```
#include <stem.h>
```

### 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.*

#### 7.27.1 Detailed Description

Class representing a stemming algorithm.

#### 7.27.2 Constructor & Destructor Documentation

##### 7.27.2.1 Xapian::Stem::Stem (const Stem & o)

Copy constructor.

### 7.27.2.2 Xapian::Stem::Stem ()

Construct a [Xapian::Stem](#) object which doesn't change terms.

Equivalent to [Stem](#)("none").

### 7.27.2.3 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)
- italian (it)
- norwegian (no)
- portuguese (pt)
- russian (ru)
- spanish (es)
- swedish (sv)

#### Exceptions:

*Xapian::InvalidArgumentError* is thrown if language isn't recognised.

### 7.27.2.4 Xapian::Stem::~~Stem ()

Destructor.

## 7.27.3 Member Function Documentation

### 7.27.3.1 void Xapian::Stem::operator= (const Stem & o)

Assignment.

### 7.27.3.2 std::string Xapian::Stem::operator() (const std::string & word) const

[Stem](#) a word.

#### Parameters:

*word* a word to stem.

#### Returns:

the stem

### 7.27.3.3 std::string Xapian::Stem::get\_description () const

Return a string describing this object.

### 7.27.3.4 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.

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

- include/xapian/[stem.h](#)

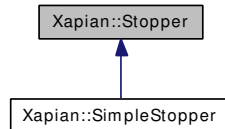


## 7.28 Xapian::Stopper Class Reference

Base class for stop-word decision functor.

```
#include <queryparser.h>
```

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.*

### 7.28.1 Detailed Description

Base class for stop-word decision functor.

### 7.28.2 Constructor & Destructor Documentation

**7.28.2.1** virtual Xapian::Stopper::~~Stopper () [inline, virtual]

Class has virtual methods, so provide a virtual destructor.

### 7.28.3 Member Function Documentation

**7.28.3.1** virtual bool Xapian::Stopper::operator() (const std::string & term)  
const [pure virtual]

Is term a stop-word?

Implemented in [Xapian::SimpleStopper](#).

### 7.28.3.2 `virtual std::string Xapian::Stopper::get_description () const` [virtual]

Return a string describing this object.

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

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

- `include/xapian/queryparser.h`

## 7.29 Xapian::StringAndFrequency Struct Reference

A string with a corresponding frequency.

```
#include <matchspy.h>
```

### Public Member Functions

- **StringAndFrequency** (std::string str\_, [Xapian::doccount](#) frequency\_)

### Public Attributes

- std::string **str**
- [Xapian::doccount](#) **frequency**

#### 7.29.1 Detailed Description

A string with a corresponding frequency.

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

- include/xapian/[matchspy.h](#)

## 7.30 Xapian::StringListSerialiser Class Reference

Class to serialise a list of strings in a form suitable for [ValueCountMatchSpy](#).

```
#include <matchspy.h>
```

### Public Member Functions

- [StringListSerialiser](#) ()  
*Default constructor.*
- [StringListSerialiser](#) (const std::string &initial)  
*Initialise with a string.*
- template<class Iterator>  
[StringListSerialiser](#) (Iterator begin, Iterator end)  
*Initialise from a pair of iterators.*
- void [append](#) (const std::string &value)  
*Add a string to the end of the list.*
- const std::string & [get](#) () const  
*Get the serialised result.*

### 7.30.1 Detailed Description

Class to serialise a list of strings in a form suitable for [ValueCountMatchSpy](#).

### 7.30.2 Constructor & Destructor Documentation

#### 7.30.2.1 Xapian::StringListSerialiser::StringListSerialiser () [inline]

Default constructor.

#### 7.30.2.2 Xapian::StringListSerialiser::StringListSerialiser (const std::string &initial) [inline]

Initialise with a string.

(The string represents a serialised form, rather than a single value to be serialised.)

**7.30.2.3** `template<class Iterator>  
Xapian::StringListSerialiser::StringListSerialiser  
(Iterator begin, Iterator end) [inline]`

Initialise from a pair of iterators.

### 7.30.3 Member Function Documentation

**7.30.3.1** `void Xapian::StringListSerialiser::append (const std::string & value)`

Add a string to the end of the list.

**7.30.3.2** `const std::string& Xapian::StringListSerialiser::get () const  
[inline]`

Get the serialised result.

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

- include/xapian/[matchspy.h](#)

## 7.31 Xapian::StringListUnserialiser Class Reference

Class to unserialise a list of strings serialised by a [StringListSerialiser](#).

```
#include <matchspy.h>
```

### Public Types

- typedef std::input\_iterator\_tag **iterator\_category**
- typedef std::string **value\_type**
- typedef size\_t **difference\_type**
- typedef std::string \* **pointer**
- typedef std::string & **reference**

### Public Member Functions

- [StringListUnserialiser](#) ()  
*Default constructor - use this to define an end iterator.*
- [StringListUnserialiser](#) (const std::string &in)  
*Constructor which takes a serialised list of strings, and creates an iterator pointing to the first of them.*
- [~StringListUnserialiser](#) ()  
*Destructor - nothing special to release.*
- [StringListUnserialiser](#) (const [StringListUnserialiser](#) &other)  
*Copy constructor.*
- void **operator=** (const [StringListUnserialiser](#) &other)  
*Assignment operator.*
- std::string **operator \*** () const  
*Get the current item.*
- [StringListUnserialiser](#) & **operator++** ()  
*Move to the next item.*
- [StringListUnserialiser](#) **operator++** (int)  
*Move to the next item (postfix).*

## Friends

- bool **operator==** (const [StringListUnserialiser](#) &a, const [StringListUnserialiser](#) &b)  
*Compare this iterator with another.*
- bool **operator!=** (const [StringListUnserialiser](#) &a, const [StringListUnserialiser](#) &b)

### 7.31.1 Detailed Description

Class to unserialise a list of strings serialised by a [StringListSerialiser](#).

The class can be used as an iterator: use the default constructor to get an end iterator.

### 7.31.2 Constructor & Destructor Documentation

#### 7.31.2.1 Xapian::StringListUnserialiser::StringListUnserialiser () [inline]

Default constructor - use this to define an end iterator.

#### 7.31.2.2 Xapian::StringListUnserialiser::StringListUnserialiser (const std::string &in) [inline]

Constructor which takes a serialised list of strings, and creates an iterator pointing to the first of them.

#### 7.31.2.3 Xapian::StringListUnserialiser::~~StringListUnserialiser () [inline]

Destructor - nothing special to release.

#### 7.31.2.4 Xapian::StringListUnserialiser::StringListUnserialiser (const StringListUnserialiser &other) [inline]

Copy constructor.

### 7.31.3 Member Function Documentation

#### 7.31.3.1 void Xapian::StringListUnserialiser::operator= (const StringListUnserialiser &other) [inline]

Assignment operator.

**7.31.3.2** `std::string Xapian::StringListUnserialiser::operator * () const`  
[inline]

Get the current item.

**7.31.3.3** `StringListUnserialiser& Xapian::StringListUnserialiser::operator++()` [inline]

Move to the next item.

**7.31.3.4** `StringListUnserialiser Xapian::StringListUnserialiser::operator++(int)` [inline]

Move to the next item (postfix).

## 7.31.4 Friends And Related Function Documentation

**7.31.4.1** `bool operator== (const StringListUnserialiser & a, const StringListUnserialiser & b)` [friend]

Compare this iterator with another.

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

- include/xapian/[matchspy.h](#)

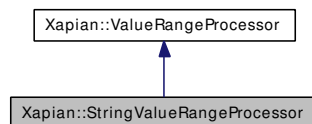


## 7.32 Xapian::StringValueRangeProcessor Class Reference

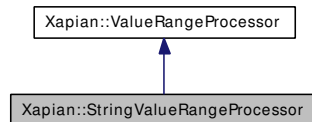
Handle a string range.

```
#include <queryparser.h>
```

Inheritance diagram for Xapian::StringValueRangeProcessor:



Collaboration 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.*

#### 7.32.1 Detailed Description

Handle a string range.

The end points can be any strings.

#### 7.32.2 Constructor & Destructor Documentation

##### 7.32.2.1 Xapian::StringValueRangeProcessor::StringValueRangeProcessor (Xapian::valueno valno\_) [inline]

Constructor.

**Parameters:**

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

### 7.32.3 Member Function Documentation

#### 7.32.3.1 `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:

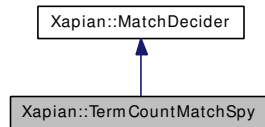
- `include/xapian/queryparser.h`

## 7.33 Xapian::TermCountMatchSpy Class Reference

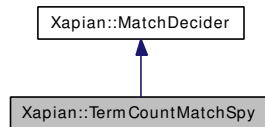
Class for counting the frequencies of terms in the matching documents.

```
#include <matchspy.h>
```

Inheritance diagram for Xapian::TermCountMatchSpy:



Collaboration diagram for Xapian::TermCountMatchSpy:



### Public Member Functions

- [TermCountMatchSpy](#) ()  
*Default constructor.*
- [TermCountMatchSpy](#) (std::string prefix)  
*Construct a MatchSpy which counts the terms with a particular prefix.*
- void [add\\_prefix](#) (std::string prefix)  
*Add a prefix to count terms with.*
- const std::map< std::string, [Xapian::doccount](#) > & [get\\_terms](#) (std::string prefix) const  
*Return the suffixes of those terms seen with prefix prefix.*
- size\_t [get\\_documents\\_seen](#) () const  
*Return the number of documents tallied.*
- size\_t [get\\_terms\\_seen](#) () const  
*Return the number of term occurrences tallied.*
- void [get\\_top\\_terms](#) (std::vector< [StringAndFrequency](#) > &result, std::string prefix, size\_t maxterms) const  
*Get the most frequent terms with a given prefix.*

- `bool operator() (const Xapian::Document &doc) const`  
*Implementation of virtual operator().*

## Protected Attributes

- `Xapian::doccount documents_seen`  
*Total number of documents seen by the match spy.*
- `Xapian::termcount terms_seen`  
*Total number of term instances seen by the match spy.*
- `std::map< std::string, std::map< std::string, Xapian::doccount > > terms`  
*Set of term suffixes seen for each prefix so far, together with their frequency.*

### 7.33.1 Detailed Description

Class for counting the frequencies of terms in the matching documents.

Note that accessing the list of terms is generally more expensive than accessing a value, so if it is possible to store the information you need in a value, you should probably use a [ValueCountMatchSpy](#) instead of a [TermCountMatchSpy](#).

### 7.33.2 Constructor & Destructor Documentation

#### 7.33.2.1 `Xapian::TermCountMatchSpy::TermCountMatchSpy ()` `[inline]`

Default constructor.

#### 7.33.2.2 `Xapian::TermCountMatchSpy::TermCountMatchSpy (std::string prefix)` `[inline]`

Construct a MatchSpy which counts the terms with a particular prefix.

Further prefixes can be added by calling [add\\_prefix\(\)](#).

### 7.33.3 Member Function Documentation

#### 7.33.3.1 `void Xapian::TermCountMatchSpy::add_prefix (std::string prefix)` `[inline]`

Add a prefix to count terms with.

A [TermCountMatchSpy](#) can count terms with one or more prefixes. If the prefixes overlap (eg, "X" and "XA"), terms with both prefixes will be counted for each of those prefixes.

**7.33.3.2** `const std::map<std::string, Xapian::doccount>&  
Xapian::TermCountMatchSpy::get_terms (std::string prefix) const  
[inline]`

Return the suffixes of those terms seen with prefix *prefix*.

**Parameters:**

*prefix* The prefix to examine (must have specified for examination before performing the match - either by using the [add\\_prefix\(\)](#) method, or using the constructor which takes a prefix.)

**7.33.3.3** `size_t Xapian::TermCountMatchSpy::get_documents_seen () const  
[inline]`

Return the number of documents tallied.

**7.33.3.4** `size_t Xapian::TermCountMatchSpy::get_terms_seen () const  
[inline]`

Return the number of term occurrences tallied.

If terms occur in more than one of the prefixes specified, they will be counted multiple times.

**7.33.3.5** `void Xapian::TermCountMatchSpy::get_top_terms (std::vector<  
StringAndFrequency > & result, std::string prefix, size_t maxterms)  
const`

Get the most frequent terms with a given prefix.

**Parameters:**

*result* A vector which will be filled with the most frequent terms, in descending order of frequency. Terms with the same frequency will be sorted in ascending alphabetical order.

*prefix* The prefix to examine (must have specified for examination before performing the match - either by using the [add\\_prefix\(\)](#) method, or using the constructor which takes a prefix.)

*maxterms* The maximum number of terms to return.

**7.33.3.6** `bool Xapian::TermCountMatchSpy::operator() (const  
Xapian::Document & doc) const [virtual]`

Implementation of virtual operator().

This implementation tallies terms for a matching document.

Implements [Xapian::MatchDecider](#).

## 7.33.4 Member Data Documentation

### 7.33.4.1 **Xapian::doccount Xapian::TermCountMatchSpy::documents\_seen** [mutable, protected]

Total number of documents seen by the match spy.

### 7.33.4.2 **Xapian::termcount Xapian::TermCountMatchSpy::terms\_seen** [mutable, protected]

Total number of term instances seen by the match spy.

### 7.33.4.3 **std::map<std::string, std::map<std::string, Xapian::doccount> >** **Xapian::TermCountMatchSpy::terms** [mutable, protected]

Set of term suffixes seen for each prefix so far, together with their frequency.

Only the suffix (ie, the part of the term after the prefix) is stored, to reduce memory usage.

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

- include/xapian/[matchspy.h](#)

## 7.34 Xapian::TermGenerator Class Reference

Parses a piece of text and generate terms.

```
#include <termgenerator.h>
```

### 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.*

### 7.34.1 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.

### 7.34.2 Member Enumeration Documentation

#### 7.34.2.1 enum [Xapian::TermGenerator::flags](#)

Flags to OR together and pass to [TermGenerator::set\\_flags\(\)](#).

**Enumerator:**

***FLAG\_SPELLING*** Index data required for spelling correction.



### 7.34.3 Constructor & Destructor Documentation

#### 7.34.3.1 Xapian::TermGenerator::TermGenerator (const TermGenerator & o)

Copy constructor.

#### 7.34.3.2 Xapian::TermGenerator::TermGenerator ()

Default constructor.

#### 7.34.3.3 Xapian::TermGenerator::~~TermGenerator ()

Destructor.

### 7.34.4 Member Function Documentation

#### 7.34.4.1 TermGenerator& Xapian::TermGenerator::operator= (const TermGenerator & o)

Assignment.

#### 7.34.4.2 void Xapian::TermGenerator::set\_stemmer (const Xapian::Stem & stemmer)

Set the [Xapian::Stem](#) object to be used for generating stemmed terms.

#### 7.34.4.3 void Xapian::TermGenerator::set\_stopper (const Xapian::Stopper \* stop = NULL)

Set the [Xapian::Stopper](#) object to be used for identifying stopwords.

#### 7.34.4.4 void Xapian::TermGenerator::set\_document (const Xapian::Document & doc)

Set the current document.

#### 7.34.4.5 const Xapian::Document& Xapian::TermGenerator::get\_document () const

Get the current document.

**7.34.4.6 void Xapian::TermGenerator::set\_database (const Xapian::WritableDatabase & db)**

Set the database to index spelling data to.

**7.34.4.7 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.

**7.34.4.8 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).

**7.34.4.9 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).

**7.34.4.10** `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.

**7.34.4.11** `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.

**7.34.4.12** `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).

**7.34.4.13** `Xapian::termcount Xapian::TermGenerator::get_termpos () const`

Get the current term position.

**7.34.4.14** `void Xapian::TermGenerator::set_termpos (Xapian::termcount termpos)`

Set the current term position.

**7.34.4.15** `std::string Xapian::TermGenerator::get_description () const`

Return a string describing this object.

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

- `include/xapian/termgenerator.h`

## 7.35 Xapian::TermIterator Class Reference

An iterator pointing to items in a list of terms.

```
#include <termiterator.h>
```

### Public Types

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

### 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 &tname)  
*Skip the iterator to term tname, or the first term after tname if tname isn't in the list of terms being iterated.*
- [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::RefCntPtr< Internal > **internal**

### 7.35.1 Detailed Description

An iterator pointing to items in a list of terms.

### 7.35.2 Member Typedef Documentation

#### 7.35.2.1 `typedef std::input_iterator_tag Xapian::TermIterator::iterator_category`

Allow use as an STL iterator.

### 7.35.3 Constructor & Destructor Documentation

#### 7.35.3.1 `Xapian::TermIterator::TermIterator ()`

Default constructor - for declaring an uninitialised iterator.

#### 7.35.3.2 `Xapian::TermIterator::~~TermIterator ()`

Destructor.

#### 7.35.3.3 `Xapian::TermIterator::TermIterator (const TermIterator & other)`

Copying is allowed.

The internals are reference counted, so copying is also cheap.

## 7.35.4 Member Function Documentation

### 7.35.4.1 `void Xapian::TermIterator::operator= (const TermIterator & other)`

Assignment is allowed.

The internals are reference counted, so assignment is also cheap.

### 7.35.4.2 `std::string Xapian::TermIterator::operator * () const`

Return the current term.

### 7.35.4.3 `void Xapian::TermIterator::skip_to (const std::string & tname)`

Skip the iterator to term *tname*, or the first term after *tname* if *tname* isn't in the list of terms being iterated.

### 7.35.4.4 `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.

### 7.35.4.5 `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.

### 7.35.4.6 `Xapian::termcount Xapian::TermIterator::positionlist_count () const`

Return length of positionlist for current term.

### 7.35.4.7 `PositionIterator Xapian::TermIterator::positionlist_begin () const`

Return [PositionIterator](#) pointing to start of positionlist for current term.

### 7.35.4.8 `PositionIterator Xapian::TermIterator::positionlist_end () const` `[inline]`

Return [PositionIterator](#) pointing to end of positionlist for current term.

#### 7.35.4.9 std::string Xapian::TermIterator::get\_description () const

Return a string describing this object.

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

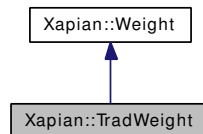
- include/xapian/[termiterator.h](#)

## 7.36 Xapian::TradWeight Class Reference

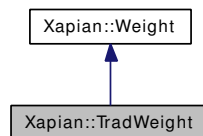
Traditional probabilistic weighting scheme.

```
#include <enquire.h>
```

Inheritance diagram for Xapian::TradWeight:



Collaboration 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*

### 7.36.1 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.

### 7.36.2 Constructor & Destructor Documentation

#### 7.36.2.1 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.

### 7.36.3 Member Function Documentation

#### 7.36.3.1 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](#).

#### 7.36.3.2 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](#).

#### 7.36.3.3 std::string Xapian::TradWeight::serialise () const [virtual]

Serialise object parameters into a string.

Implements [Xapian::Weight](#).

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

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

Implements [Xapian::Weight](#).

#### 7.36.3.5 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](#).

### 7.36.3.6 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](#).

### 7.36.3.7 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](#).

### 7.36.3.8 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](#).

### 7.36.3.9 bool Xapian::TradWeight::get\_sumpart\_needs\_doclength () const [virtual]

return false if the weight object doesn't need doclength

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

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

- include/xapian/[enquire.h](#)

## 7.37 Xapian::Utf8Iterator Class Reference

An iterator which returns unicode character values from a UTF-8 encoded string.

```
#include <unicode.h>
```

### Public Types

- typedef std::input\_iterator\_tag **iterator\_category**  
*We implement the semantics of an STL input\_iterator.*
- typedef unsigned **value\_type**
- typedef size\_t **difference\_type**
- typedef const unsigned \* **pointer**
- typedef const unsigned & **reference**

### 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.*

### 7.37.1 Detailed Description

An iterator which returns unicode character values from a UTF-8 encoded string.

### 7.37.2 Member Typedef Documentation

#### 7.37.2.1 `typedef std::input_iterator_tag Xapian::Utf8Iterator::iterator_category`

We implement the semantics of an STL `input_iterator`.

### 7.37.3 Constructor & Destructor Documentation

#### 7.37.3.1 `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.

#### 7.37.3.2 `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.

**7.37.3.3 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.

**7.37.3.4 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.

**7.37.4 Member Function Documentation****7.37.4.1 const char\* Xapian::Utf8Iterator::raw () const [inline]**

Return the raw const char \* pointer for the current position.

**7.37.4.2 size\_t Xapian::Utf8Iterator::left () const [inline]**

Return the number of bytes left in the iterator's buffer.

**7.37.4.3 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.

**7.37.4.4 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.

**7.37.4.5 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.

**7.37.4.6 Utf8Iterator Xapian::Utf8Iterator::operator++ (int) [inline]**

Move forward to the next unicode character.

**Returns:**

- An iterator pointing to the position before the move.

**7.37.4.7 Utf8Iterator& Xapian::Utf8Iterator::operator++ () [inline]**

Move forward to the next unicode character.

**Returns:**

- A reference to this object.

**7.37.4.8 bool Xapian::Utf8Iterator::operator== (const Utf8Iterator & other) const [inline]**

Test two Utf8Iterators for equality.

**Returns:**

- true iff the iterators point to the same position.

**7.37.4.9** `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.

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

- `include/xapian/unicode.h`

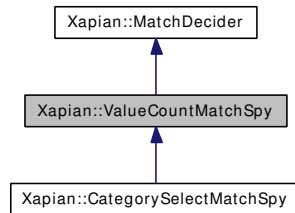


## 7.38 Xapian::ValueCountMatchSpy Class Reference

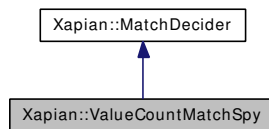
Class for counting the frequencies of values in the matching documents.

```
#include <matchspy.h>
```

Inheritance diagram for Xapian::ValueCountMatchSpy:



Collaboration diagram for Xapian::ValueCountMatchSpy:



### Public Member Functions

- [ValueCountMatchSpy](#) ()  
*Default constructor.*
- [ValueCountMatchSpy](#) ([Xapian::valueno](#) valno, bool multivalue=false)  
*Construct a MatchSpy which counts the values in a particular slot.*
- void [add\\_slot](#) ([Xapian::valueno](#) valno, bool multivalue=false)  
*Add a slot number to count values in.*
- const std::map< std::string, [Xapian::doccount](#) > & [get\\_values](#) ([Xapian::valueno](#) valno) const  
*Return the values seen in slot number valno.*
- size\_t [get\\_total](#) () const  
*Return the total number of documents tallied.*
- void [get\\_top\\_values](#) (std::vector< [StringAndFrequency](#) > &result, [Xapian::valueno](#) valno, size\_t maxvalues) const  
*Get the most frequent values in a slot.*
- bool [operator\(\)](#) (const [Xapian::Document](#) &doc) const

*Implementation of virtual operator().*

## Protected Attributes

- [Xapian::doccount total](#)

*Total number of documents seen by the match spy.*

- `std::map< Xapian::valueno, std::map< std::string, Xapian::doccount > > values`

*Set of values seen in each slot so far, together with their frequency.*

- `std::set< Xapian::valueno > multivalues`

*Set tracking which value slots can have multiple values.*

### 7.38.1 Detailed Description

Class for counting the frequencies of values in the matching documents.

### 7.38.2 Constructor & Destructor Documentation

#### 7.38.2.1 `Xapian::ValueCountMatchSpy::ValueCountMatchSpy ()` `[inline]`

Default constructor.

#### 7.38.2.2 `Xapian::ValueCountMatchSpy::ValueCountMatchSpy (Xapian::valueno valno, bool multivalue = false)` `[inline]`

Construct a MatchSpy which counts the values in a particular slot.

Further slots can be added by calling [add\\_slot\(\)](#).

### 7.38.3 Member Function Documentation

#### 7.38.3.1 `void Xapian::ValueCountMatchSpy::add_slot (Xapian::valueno valno, bool multivalue = false)` `[inline]`

Add a slot number to count values in.

A [ValueCountMatchSpy](#) can count values in one or more slots.

**7.38.3.2** `const std::map<std::string, Xapian::doccount>&  
Xapian::ValueCountMatchSpy::get_values (Xapian::valueno valno)  
const` `[inline]`

Return the values seen in slot number *valno*.

**Parameters:**

*valno* The slot to examine (must have specified for examination before performing the match - either by using the [add\\_slot\(\)](#) method, or using the constructor which takes a slot number.)

**7.38.3.3** `size_t Xapian::ValueCountMatchSpy::get_total () const` `[inline]`

Return the total number of documents tallied.

**7.38.3.4** `void Xapian::ValueCountMatchSpy::get_top_values (std::vector<  
StringAndFrequency > &result, Xapian::valueno valno, size_t  
maxvalues) const`

Get the most frequent values in a slot.

**Parameters:**

*result* A vector which will be filled with the most frequent values, in descending order of frequency. Values with the same frequency will be sorted in ascending alphabetical order.

*valno* The slot to examine (must have specified for examination before performing the match - either by using the [add\\_slot\(\)](#) method, or using the constructor which takes a slot number.)

*maxvalues* The maximum number of values to return.

**7.38.3.5** `bool Xapian::ValueCountMatchSpy::operator() (const  
Xapian::Document &doc) const` `[virtual]`

Implementation of virtual operator().

This implementation tallies values for a matching document.

Implements [Xapian::MatchDecider](#).

## 7.38.4 Member Data Documentation

**7.38.4.1** `Xapian::doccount Xapian::ValueCountMatchSpy::total` `[mutable,  
protected]`

Total number of documents seen by the match spy.

**7.38.4.2** `std::map<Xapian::valueno, std::map<std::string, Xapian::doccount>  
> Xapian::ValueCountMatchSpy::values` [mutable,  
protected]

Set of values seen in each slot so far, together with their frequency.

**7.38.4.3** `std::set<Xapian::valueno>  
Xapian::ValueCountMatchSpy::multivalues`  
[protected]

Set tracking which value slots can have multiple values.

If a valueno is in this set, its value is assumed to have been serialised by a [StringListSerialiser](#) class.

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

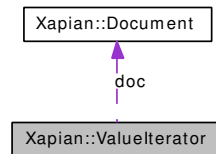
- include/xapian/[matchspy.h](#)

## 7.39 Xapian::ValueIterator Class Reference

An iterator pointing to values associated with a document.

```
#include <valueiterator.h>
```

Collaboration diagram for Xapian::ValueIterator:



### Public Types

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

### 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::value` `get_value` () const  
*Get the number of the value at the current position.*
- `std::string` `get_description` () const  
*Return a string describing this object.*

## Friends

- class `Document`
- `bool` `operator==` (const `ValueIterator` &a, const `ValueIterator` &b)
- `bool` `operator!=` (const `ValueIterator` &a, const `ValueIterator` &b)

### 7.39.1 Detailed Description

An iterator pointing to values associated with a document.

### 7.39.2 Member Typedef Documentation

#### 7.39.2.1 `typedef std::input_iterator_tag Xapian::ValueIterator::iterator_category`

Allow use as an STL iterator.

### 7.39.3 Constructor & Destructor Documentation

#### 7.39.3.1 `Xapian::ValueIterator::ValueIterator ()` `[inline]`

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

#### 7.39.3.2 `Xapian::ValueIterator::ValueIterator (const ValueIterator & other)` `[inline]`

Copying is allowed (and is cheap).

### 7.39.4 Member Function Documentation

#### 7.39.4.1 `void Xapian::ValueIterator::operator= (const ValueIterator & other)` `[inline]`

Assignment is allowed (and is cheap).

**7.39.4.2 ValueIterator& Xapian::ValueIterator::operator++ () [inline]**

Advance the iterator.

**7.39.4.3 ValueIterator Xapian::ValueIterator::operator++ (int) [inline]**

Advance the iterator (postfix variant).

**7.39.4.4 const std::string& Xapian::ValueIterator::operator \* () const**

Get the value for the current position.

**7.39.4.5 const std::string\* Xapian::ValueIterator::operator → () const**

Get the value for the current position.

**7.39.4.6 Xapian::valueno Xapian::ValueIterator::get\_valueno () const**

Get the number of the value at the current position.

**7.39.4.7 std::string Xapian::ValueIterator::get\_description () const**

Return a string describing this object.

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

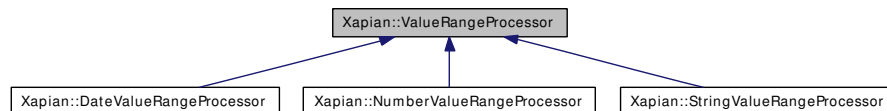
- include/xapian/[valueiterator.h](#)

## 7.40 Xapian::ValueRangeProcessor Struct Reference

Base class for value range processors.

```
#include <queryparser.h>
```

Inheritance diagram for Xapian::ValueRangeProcessor:



### Public Member Functions

- virtual [~ValueRangeProcessor](#) ()  
*Destructor.*
- virtual [Xapian::valueno operator\(\)](#) (std::string &begin, std::string &end)=0  
*See if <begin>.*

#### 7.40.1 Detailed Description

Base class for value range processors.

#### 7.40.2 Constructor & Destructor Documentation

**7.40.2.1** virtual [Xapian::ValueRangeProcessor::~~ValueRangeProcessor](#) ()  
[virtual]

Destructor.

#### 7.40.3 Member Function Documentation

**7.40.3.1** virtual [Xapian::valueno Xapian::ValueRangeProcessor::operator\(\)](#)  
(std::string & *begin*, std::string & *end*) [pure virtual]

See if <begin>.

.<end> is a valid value range.

If this [ValueRangeProcessor](#) recognises <begin>..<end> it returns the value number of 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:

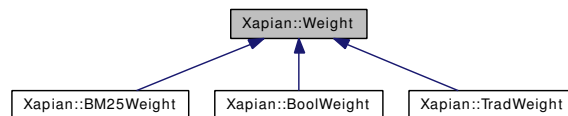
- `include/xapian/queryparser.h`

## 7.41 Xapian::Weight Class Reference

Abstract base class for weighting schemes.

```
#include <enquire.h>
```

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**

## Friends

- class **Enquire**
- class **::RemoteServer**
- class **::ScaleWeight**

### 7.41.1 Detailed Description

Abstract base class for weighting schemes.

### 7.41.2 Member Function Documentation

#### 7.41.2.1 **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.

#### 7.41.2.2 **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](#).

#### 7.41.2.3 `virtual std::string Xapian::Weight::serialise () const` [pure virtual]

Serialise object parameters into a string.

Implemented in [Xapian::BoolWeight](#), [Xapian::BM25Weight](#), and [Xapian::TradWeight](#).

#### 7.41.2.4 `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](#).

#### 7.41.2.5 `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](#).

#### 7.41.2.6 `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](#).

#### 7.41.2.7 `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](#).

**7.41.2.8 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](#).

**7.41.2.9 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](#).

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

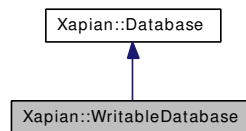
- [include/xapian/enquire.h](#)

## 7.42 Xapian::WritableDatabase Class Reference

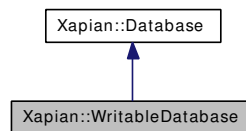
This class provides read/write access to a database.

```
#include <database.h>
```

Inheritance diagram for Xapian::WritableDatabase:



Collaboration 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](#) ()  
*Flush to disk any 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 potential 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.*

### 7.42.1 Detailed Description

This class provides read/write access to a database.

### 7.42.2 Constructor & Destructor Documentation

#### 7.42.2.1 `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.

#### 7.42.2.2 `Xapian::WritableDatabase::WritableDatabase ()`

Create an empty [WritableDatabase](#).

#### 7.42.2.3 `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

#### 7.42.2.4 `Xapian::WritableDatabase::WritableDatabase (const WritableDatabase & other)`

Copying is allowed.

The internals are reference counted, so copying is cheap.



### 7.42.3 Member Function Documentation

#### 7.42.3.1 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.

#### 7.42.3.2 void Xapian::WritableDatabase::flush ()

Flush to disk any 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 have a severe performance cost.

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.

*Xapian::DatabaseLockError* will be thrown if a lock couldn't be acquired on the database.

### 7.42.3.3 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 this flushing is that small transactions cause modifications to be frequently flushed which 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.

### 7.42.3.4 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.

#### 7.42.3.5 void Xapian::WritableDatabase::cancel\_transaction ()

Abort the transaction currently in progress, discarding the potential 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.

#### 7.42.3.6 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.

**7.42.3.7 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.

**7.42.3.8 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.

### 7.42.3.9 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.

### 7.42.3.10 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](#).

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).

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.

**7.42.3.11 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).

**7.42.3.12 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).

**7.42.3.13 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.

**7.42.3.14 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.

**7.42.3.15 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.

**7.42.3.16 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.

**7.42.3.17** `std::string Xapian::WritableDatabase::get_description () const`  
[virtual]

Return a string describing this object.

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

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

- include/xapian/[database.h](#)



## Chapter 8

# xapian-core File Documentation

### 8.1 include/xapian.h File Reference

Public interfaces for the [Xapian](#) library.

```
#include <xapian/version.h>
#include <xapian/types.h>
#include <xapian/error.h>
#include <xapian/errorhandler.h>
#include <xapian/database.h>
#include <xapian/dbfactory.h>
#include <xapian/document.h>
#include <xapian/positioniterator.h>
#include <xapian/postingiterator.h>
#include <xapian/termiterator.h>
#include <xapian/valueiterator.h>
#include <xapian/termgenerator.h>
#include <xapian/enquire.h>
#include <xapian/expanddecider.h>
#include <xapian/query.h>
#include <xapian/queryparser.h>
#include <xapian/sorter.h>
#include <xapian/stem.h>
#include <xapian/unicode.h>
#include <xapian/visibility.h>
```

```
#include <xapian/deprecated.h>
```

## Namespaces

- namespace [Xapian](#)

## Functions

- XAPIAN\_VISIBILITY\_DEFAULT const char \* [Xapian::version\\_string](#) ()  
*Report the version string of the library which the program is linked with.*
- XAPIAN\_VISIBILITY\_DEFAULT [Xapian::XAPIAN\\_DEPRECATED](#) (const char \*xapian\_version\_string())  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*
- XAPIAN\_VISIBILITY\_DEFAULT int [Xapian::major\\_version](#) ()  
*Report the major version of the library which the program is linked to.*
- XAPIAN\_VISIBILITY\_DEFAULT [Xapian::XAPIAN\\_DEPRECATED](#) (int xapian\_major\_version())  
*For compatibility with [Xapian](#) 0.9.5 and earlier.*
- XAPIAN\_VISIBILITY\_DEFAULT int [Xapian::minor\\_version](#) ()  
*Report the minor version of the library which the program is linked to.*
- XAPIAN\_VISIBILITY\_DEFAULT int [Xapian::revision](#) ()  
*Report the revision of the library which the program is linked to.*

### 8.1.1 Detailed Description

Public interfaces for the [Xapian](#) library.

## 8.2 include/xapian/database.h File Reference

API for working with [Xapian](#) databases.

```
#include <string>
#include <vector>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
#include <xapian/postingiterator.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### 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.*

### 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.*

### 8.2.1 Detailed Description

API for working with [Xapian](#) databases.

## 8.3 include/xapian/dbfactory.h File Reference

Factory functions for constructing Database and WritableDatabase objects.

```
#include <string>
#include <xapian/types.h>
#include <xapian/deprecated.h>
#include <xapian/version.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)
- namespace **Xapian::Auto**
- namespace **Xapian::InMemory**
- namespace **Xapian::Quartz**
- namespace **Xapian::Flint**
- namespace **Xapian::Remote**

### Functions

- XAPIAN\_VISIBILITY\_DEFAULT Database [Xapian::Auto::open\\_stub](#) (const std::string &file)  
*Construct a [Database](#) object for a stub database file.*
- XAPIAN\_VISIBILITY\_DEFAULT WritableDatabase [Xapian::InMemory::open](#) ()  
*Construct a [WritableDatabase](#) object for a new, empty InMemory database.*
- XAPIAN\_VISIBILITY\_DEFAULT [Xapian::Quartz::XAPIAN\\_DEPRECATED](#) (Database open(const std::string &dir))  
*Construct a [Database](#) object for read-only access to a Quartz database.*
- XAPIAN\_VISIBILITY\_DEFAULT [Xapian::Quartz::XAPIAN\\_DEPRECATED](#) (WritableDatabase open(const std::string &dir, int action, int block\_size=8192))  
*Construct a [Database](#) object for update access to a Quartz database.*
- XAPIAN\_VISIBILITY\_DEFAULT Database [Xapian::Flint::open](#) (const std::string &dir)  
*Construct a [Database](#) object for read-only access to a Flint database.*
- XAPIAN\_VISIBILITY\_DEFAULT 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.*

- XAPIAN\_VISIBILITY\_DEFAULT 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.*

- XAPIAN\_VISIBILITY\_DEFAULT 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.*

- XAPIAN\_VISIBILITY\_DEFAULT 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.*

- XAPIAN\_VISIBILITY\_DEFAULT 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.*

### 8.3.1 Detailed Description

Factory functions for constructing Database and WritableDatabase objects.

## 8.4 include/xapian/document.h File Reference

API for working with documents.

```
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::Document](#)  
*A document in the database - holds data, values, terms, and postings.*

### 8.4.1 Detailed Description

API for working with documents.

## 8.5 include/xapian/enquire.h File Reference

API for running queries.

```
#include <string>
#include <xapian/base.h>
#include <xapian/deprecated.h>
#include <xapian/sorter.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### 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.*

## Functions

- bool **Xapian::operator==** (const MSetIterator &a, const MSetIterator &b)
- bool **Xapian::operator!=** (const MSetIterator &a, const MSetIterator &b)
- bool **Xapian::operator==** (const ESetIterator &a, const ESetIterator &b)
- bool **Xapian::operator!=** (const ESetIterator &a, const ESetIterator &b)

### 8.5.1 Detailed Description

API for running queries.

## 8.6 include/xapian/errorhandler.h File Reference

Decide if a Xapian::Error exception should be ignored.

```
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::ErrorHandler](#)

*Decide if a Xapian::Error exception should be ignored.*

### 8.6.1 Detailed Description

Decide if a Xapian::Error exception should be ignored.

## 8.7 include/xapian/expanddecider.h File Reference

Allow rejection of terms during ESet generation.

```
#include <set>
#include <string>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### 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.*

### 8.7.1 Detailed Description

Allow rejection of terms during ESet generation.

## 8.8 include/xapian/matchspy.h File Reference

MatchDecider subclasses for use as "match spies".

```
#include <xapian/enquire.h>
#include <map>
#include <set>
#include <string>
#include <vector>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::MultipleMatchDecider](#)  
*Class which applies several match deciders in turn.*
- struct [Xapian::StringAndFrequency](#)  
*A string with a corresponding frequency.*
- class [Xapian::StringListSerialiser](#)  
*Class to serialise a list of strings in a form suitable for [ValueCountMatchSpy](#).*
- class [Xapian::StringListUnserialiser](#)  
*Class to unserialise a list of strings serialised by a [StringListSerialiser](#).*
- class [Xapian::ValueCountMatchSpy](#)  
*Class for counting the frequencies of values in the matching documents.*
- class [Xapian::TermCountMatchSpy](#)  
*Class for counting the frequencies of terms in the matching documents.*
- class [Xapian::CategorySelectMatchSpy](#)  
*MatchSpy for classifying matching documents by their values.*

### Functions

- bool **Xapian::operator==** (const StringListUnserialiser &a, const StringListUnserialiser &b)
- bool **Xapian::operator!=** (const StringListUnserialiser &a, const StringListUnserialiser &b)

### **8.8.1 Detailed Description**

MatchDecider subclasses for use as "match spies".

## 8.9 include/xapian/positioniterator.h File Reference

Classes for iterating through position lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class **Xapian::TermPosWrapper**
- class [Xapian::PositionIterator](#)

*An iterator pointing to items in a list of positions.*

### 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.*

#### 8.9.1 Detailed Description

Classes for iterating through position lists.

## 8.10 include/xapian/postingiterator.h File Reference

Classes for iterating through posting lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::DocIDWrapper](#)
- class [Xapian::PostingIterator](#)

*An iterator pointing to items in a list of postings.*

### 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.*

#### 8.10.1 Detailed Description

Classes for iterating through posting lists.

## 8.11 include/xapian/query.h File Reference

Classes for representing a query.

```
#include <string>
#include <vector>
#include <xapian/base.h>
#include <xapian/deprecated.h>
#include <xapian/types.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::Query](#)  
*Class representing a query.*
- class [Xapian::Query](#)  
*Class representing a query.*

### 8.11.1 Detailed Description

Classes for representing a query.



## 8.12 include/xapian/queryparser.h File Reference

parsing a user query string to build a [Xapian::Query](#) object

```
#include <xapian/base.h>
#include <xapian/query.h>
#include <xapian/termiterator.h>
#include <xapian/visibility.h>
#include <set>
#include <string>
```

### Namespaces

- namespace [Xapian](#)

### 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.*

### Functions

- XAPIAN\_VISIBILITY\_DEFAULT std::string [Xapian::sortable\\_serialise](#) (double value)  
*Convert a floating point number to a string, preserving sort order.*

- XAPIAN\_VISIBILITY\_DEFAULT double [Xapian::sortable\\_unserialise](#) (const std::string &value)

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

### 8.12.1 Detailed Description

parsing a user query string to build a [Xapian::Query](#) object

## 8.13 include/xapian/sorter.h File Reference

Build sort keys for MSet ordering.

```
#include <string>
#include <vector>
#include <xapian/document.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::Sorter](#)  
*Virtual base class for sorter functor.*
- class [Xapian::MultiValueSorter](#)  
*[Sorter](#) subclass which sorts by a several values.*

### 8.13.1 Detailed Description

Build sort keys for MSet ordering.

## 8.14 include/xapian/stem.h File Reference

stemming algorithms

```
#include <xapian/base.h>
#include <xapian/visibility.h>
#include <string>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::Stem](#)  
*Class representing a stemming algorithm.*

#### 8.14.1 Detailed Description

stemming algorithms

## 8.15 include/xapian/termgenerator.h File Reference

parse free text and generate terms

```
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/unicode.h>
#include <xapian/visibility.h>
#include <string>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::TermGenerator](#)  
*Parses a piece of text and generate terms.*

### 8.15.1 Detailed Description

parse free text and generate terms

## 8.16 include/xapian/termiterator.h File Reference

Classes for iterating through term lists.

```
#include <iterator>
#include <string>
#include <xapian/base.h>
#include <xapian/types.h>
#include <xapian/positioniterator.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class **Xapian::TermNameWrapper**
- class [Xapian::TermIterator](#)

*An iterator pointing to items in a list of terms.*

### Functions

- bool **Xapian::operator==** (const TermIterator &a, const TermIterator &b)
- bool **Xapian::operator!=** (const TermIterator &a, const TermIterator &b)

#### 8.16.1 Detailed Description

Classes for iterating through term lists.

## 8.17 include/xapian/types.h File Reference

typedefs for [Xapian](#)

### Namespaces

- namespace [Xapian](#)

### 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 microseconds.*
- 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".*

### 8.17.1 Detailed Description

typedefs for [Xapian](#)



## 8.18 include/xapian/unicode.h File Reference

Unicode and UTF-8 related classes and functions.

```
#include <xapian/visibility.h>
```

```
#include <string>
```

### Namespaces

- namespace [Xapian](#)
- namespace **Xapian::Unicode**
- namespace **Xapian::Unicode::Internal**

### Classes

- class [Xapian::Utf8Iterator](#)

*An iterator which returns unicode character values from a UTF-8 encoded string.*

### 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 one of these categories.*

## Functions

- `XAPIAN_VISIBILITY_DEFAULT` `int` `Xapian::Unicode::Internal::get_character_info` (unsigned ch)  
*Convert a single non-ASCII unicode character to UTF-8.*
- `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)
- `XAPIAN_VISIBILITY_DEFAULT` `unsigned` `Xapian::Unicode::nonascii_to_utf8` (unsigned ch, char \*buf)  
*Convert a single 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 is a given unicode character is a letter or number.*
- `bool` `Xapian::Unicode::is_whitespace` (unsigned ch)  
*Test is a given unicode character is a whitespace character.*
- `bool` `Xapian::Unicode::is_currency` (unsigned ch)  
*Test is 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.*

### 8.18.1 Detailed Description

Unicode and UTF-8 related classes and functions.

## 8.19 include/xapian/valueiterator.h File Reference

classes for iterating through values

```
#include <iterator>
#include <string>
#include <xapian/types.h>
#include <xapian/document.h>
#include <xapian/visibility.h>
```

### Namespaces

- namespace [Xapian](#)

### Classes

- class [Xapian::ValueIterator](#)  
*An iterator pointing to values associated with a document.*

### Functions

- bool **Xapian::operator==** (const ValueIterator &a, const ValueIterator &b)
- bool **Xapian::operator!=** (const ValueIterator &a, const ValueIterator &b)

#### 8.19.1 Detailed Description

classes for iterating through values



## Chapter 9

# xapian-core Page Documentation

### 9.1 Deprecated List

**Member [Xapian::Enquire::XAPIAN\\_DEPRECATED](#)(void register\_match\_decider(const std::string &name, const M**

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::Query::XAPIAN\\_DEPRECATED](#)(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 with a parameter value.

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

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

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

**Member [Xapian::XAPIAN\\_DEPRECATED](#)(const char \*xapian\_version\_string())**

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

# Index

- ~Database
  - Xapian::Database, [38](#)
- ~Document
  - Xapian::Document, [49](#)
- ~ESet
  - Xapian::ESet, [68](#)
- ~Enquire
  - Xapian::Enquire, [56](#)
- ~ErrorHandler
  - Xapian::ErrorHandler, [65](#)
- ~ExpandDecider
  - Xapian::ExpandDecider, [73](#)
- ~Internal
  - Xapian::Query, [111](#)
- ~MSet
  - Xapian::MSet, [82](#)
- ~MatchDecider
  - Xapian::MatchDecider, [78](#)
- ~PositionIterator
  - Xapian::PositionIterator, [100](#)
- ~PostingIterator
  - Xapian::PostingIterator, [102](#)
- ~Query
  - Xapian::Query, [109](#)
- ~QueryParser
  - Xapian::QueryParser, [118](#)
- ~RSet
  - Xapian::RSet, [123](#)
- ~SimpleStopper
  - Xapian::SimpleStopper, [126](#)
- ~Sorter
  - Xapian::Sorter, [127](#)
- ~Stem
  - Xapian::Stem, [129](#)
- ~Stopper
  - Xapian::Stopper, [131](#)
- ~StringListUnserialiser
  - Xapian::StringListUnserialiser, [137](#)
- ~TermGenerator
  - Xapian::TermGenerator, [147](#)
- ~TermIterator
  - Xapian::TermIterator, [151](#)
- ~ValueRangeProcessor
  - Xapian::ValueRangeProcessor, [170](#)
- ~WritableDatabase
  - Xapian::WritableDatabase, [178](#)
- add
  - Xapian::SimpleStopper, [126](#)
- add\_boolean\_prefix
  - Xapian::QueryParser, [120](#)
- add\_database
  - Xapian::Database, [38](#)
- add\_document
  - Xapian::RSet, [123](#)
  - Xapian::WritableDatabase, [181](#)
- add\_posting
  - Xapian::Document, [50](#)
- add\_prefix
  - Xapian::QueryParser, [119](#)
  - Xapian::TermCountMatchSpy, [142](#)
- add\_slot
  - Xapian::ValueCountMatchSpy, [164](#)
- add\_spelling
  - Xapian::WritableDatabase, [184](#)
- add\_subquery
  - Xapian::Query, [113](#)
- add\_synonym
  - Xapian::WritableDatabase, [184](#)
- add\_term
  - Xapian::Document, [50](#)
- add\_value
  - Xapian::Document, [49](#)
- add\_valuerangeprocessor
  - Xapian::QueryParser, [121](#)
- allterms\_begin
  - Xapian::Database, [40](#)
- allterms\_end
  - Xapian::Database, [40](#)
- append
  - Xapian::MultipleMatchDecider, [92](#)
  - Xapian::StringListSerialiser, [135](#)

- assign
  - Xapian::Utf8Iterator, [160](#)
- back
  - Xapian::ESet, [69](#)
  - Xapian::MSet, [85](#)
- BAD\_VALUENO
  - Xapian, [21](#)
- begin
  - Xapian::ESet, [69](#)
  - Xapian::MSet, [85](#)
- begin\_transaction
  - Xapian::WritableDatabase, [179](#)
- BM25Weight
  - Xapian::BM25Weight, [25](#)
- build\_numeric\_ranges
  - Xapian::CategorySelectMatchSpy, [33](#)
- cancel\_transaction
  - Xapian::WritableDatabase, [181](#)
- CategorySelectMatchSpy
  - Xapian::CategorySelectMatchSpy, [33](#)
- clear\_synonyms
  - Xapian::WritableDatabase, [185](#)
- clear\_terms
  - Xapian::Document, [51](#)
- clear\_values
  - Xapian::Document, [49](#)
- clone
  - Xapian::BM25Weight, [25](#)
  - Xapian::BoolWeight, [29](#)
  - Xapian::TradWeight, [156](#)
- commit\_transaction
  - Xapian::WritableDatabase, [180](#)
- contains
  - Xapian::RSet, [124](#)
- convert\_to\_percent
  - Xapian::MSet, [83](#)
- create
  - Xapian::Weight, [173](#)
- Database
  - Xapian::Database, [38](#)
- DateValueRangeProcessor
  - Xapian::DateValueRangeProcessor, [45](#)
- DB\_CREATE
  - Xapian, [21](#)
- DB\_CREATE\_OR\_OPEN
  - Xapian, [21](#)
- DB\_CREATE\_OR\_OVERWRITE
  - Xapian, [21](#)
- DB\_OPEN
  - Xapian, [21](#)
- delete\_document
  - Xapian::WritableDatabase, [182](#)
- doccount
  - Xapian, [17](#)
- doccount\_diff
  - Xapian, [17](#)
- docid
  - Xapian, [17](#)
- doclength
  - Xapian, [17](#)
- Document
  - Xapian::Document, [48](#)
- documents\_seen
  - Xapian::TermCountMatchSpy, [144](#)
- empty
  - Xapian::ESet, [69](#)
  - Xapian::MSet, [85](#)
  - Xapian::Query, [112](#)
  - Xapian::RSet, [123](#)
- end
  - Xapian::ESet, [69](#)
  - Xapian::MSet, [85](#)
- end\_construction
  - Xapian::Query, [113](#)
- Enquire
  - Xapian::Enquire, [56](#)
- ErrorHandler
  - Xapian::ErrorHandler, [65](#)
- ESet
  - Xapian::ESet, [68](#)
- ESetIterator
  - Xapian::ESetIterator, [71](#)
- ExpandDeciderAnd
  - Xapian::ExpandDeciderAnd, [75](#)
- ExpandDeciderFilterTerms
  - Xapian::ExpandDeciderFilterTerms, [76](#)
- feature\_flag
  - Xapian::QueryParser, [116](#)
- fetch
  - Xapian::MSet, [82, 83](#)

- FLAG\_AUTO\_MULTIWORD\_-  
SYNONYMS  
Xapian::QueryParser, 117
- FLAG\_AUTO\_SYNONYMS  
Xapian::QueryParser, 117
- FLAG\_BOOLEAN  
Xapian::QueryParser, 117
- FLAG\_BOOLEAN\_ANY\_CASE  
Xapian::QueryParser, 117
- FLAG\_LOVEHATE  
Xapian::QueryParser, 117
- FLAG\_PARTIAL  
Xapian::QueryParser, 117
- FLAG\_PHRASE  
Xapian::QueryParser, 117
- FLAG\_PURE\_NOT  
Xapian::QueryParser, 117
- FLAG\_SPELLING  
Xapian::TermGenerator, 146
- FLAG\_SPELLING\_CORRECTION  
Xapian::QueryParser, 117
- FLAG\_SYNONYM  
Xapian::QueryParser, 117
- FLAG\_WILDCARD  
Xapian::QueryParser, 117
- flags  
Xapian::TermGenerator, 146
- flush  
Xapian::WritableDatabase, 179
- get  
Xapian::StringListSerialiser, 135
- get\_available\_languages  
Xapian::Stem, 130
- get\_avlength  
Xapian::Database, 41
- get\_collapse\_count  
Xapian::MSetIterator, 90
- get\_collapse\_key  
Xapian::MSetIterator, 90
- get\_collection\_freq  
Xapian::Database, 41
- get\_corrected\_query\_string  
Xapian::QueryParser, 121
- get\_data  
Xapian::Document, 49
- get\_default\_op  
Xapian::QueryParser, 119
- get\_description  
Xapian::Database, 39
- Xapian::Document, 52
- Xapian::Enquire, 64
- Xapian::ESet, 69
- Xapian::ESetIterator, 72
- Xapian::MSet, 86
- Xapian::MSetIterator, 91
- Xapian::PositionIterator, 100
- Xapian::PostingIterator, 103
- Xapian::Query, 112, 113
- Xapian::QueryParser, 121
- Xapian::RSet, 124
- Xapian::SimpleStopper, 126
- Xapian::Stem, 130
- Xapian::Stopper, 131
- Xapian::TermGenerator, 149
- Xapian::TermIterator, 152
- Xapian::ValueIterator, 169
- Xapian::WritableDatabase, 186
- get\_doccount  
Xapian::Database, 40
- get\_docid  
Xapian::Document, 52
- get\_doclength  
Xapian::Database, 41
- Xapian::PostingIterator, 103
- get\_document  
Xapian::Database, 41
- Xapian::MSetIterator, 89
- Xapian::TermGenerator, 147
- get\_documents\_seen  
Xapian::TermCountMatchSpy, 143
- get\_ebound  
Xapian::ESet, 68
- get\_eset  
Xapian::Enquire, 61, 62
- get\_firstitem  
Xapian::MSet, 84
- get\_lastdocid  
Xapian::Database, 41
- get\_length  
Xapian::Query, 111, 113
- get\_matches\_estimated  
Xapian::MSet, 84
- get\_matches\_lower\_bound  
Xapian::MSet, 84
- get\_matches\_upper\_bound  
Xapian::MSet, 84
- get\_matching\_terms\_begin  
Xapian::Enquire, 62, 63
- get\_matching\_terms\_end



- Xapian::Enquire, 63
- get\_max\_attained
  - Xapian::MSet, 85
- get\_max\_possible
  - Xapian::MSet, 85
- get\_maxextra
  - Xapian::BM25Weight, 26
  - Xapian::BoolWeight, 30
  - Xapian::TradWeight, 157
  - Xapian::Weight, 175
- get\_maxpart
  - Xapian::BM25Weight, 26
  - Xapian::BoolWeight, 30
  - Xapian::TradWeight, 156
  - Xapian::Weight, 174
- get\_metadata
  - Xapian::Database, 43
- get\_mset
  - Xapian::Enquire, 60
- get\_parameter
  - Xapian::Query, 113
- get\_percent
  - Xapian::MSetIterator, 90
- get\_query
  - Xapian::Enquire, 57
- get\_rank
  - Xapian::MSetIterator, 90
- get\_spelling\_suggestion
  - Xapian::Database, 42
- get\_sumextra
  - Xapian::BM25Weight, 26
  - Xapian::BoolWeight, 30
  - Xapian::TradWeight, 157
  - Xapian::Weight, 174
- get\_sumpart
  - Xapian::BM25Weight, 26
  - Xapian::BoolWeight, 29
  - Xapian::TradWeight, 156
  - Xapian::Weight, 174
- get\_sumpart\_needs\_doclength
  - Xapian::BM25Weight, 27
  - Xapian::BoolWeight, 30
  - Xapian::TradWeight, 157
  - Xapian::Weight, 175
- get\_termfreq
  - Xapian::Database, 41
  - Xapian::MSet, 83
  - Xapian::TermIterator, 152
- get\_termpos
  - Xapian::TermGenerator, 149
- get\_terms
  - Xapian::Query, 113
  - Xapian::TermCountMatchSpy, 142
- get\_terms\_begin
  - Xapian::Query, 111
- get\_terms\_end
  - Xapian::Query, 112
- get\_terms\_seen
  - Xapian::TermCountMatchSpy, 143
- get\_termweight
  - Xapian::MSet, 84
- get\_top\_terms
  - Xapian::TermCountMatchSpy, 143
- get\_top\_values
  - Xapian::ValueCountMatchSpy, 165
- get\_total
  - Xapian::ValueCountMatchSpy, 165
- get\_value
  - Xapian::Document, 49
- get\_valueno
  - Xapian::ValueIterator, 169
- get\_values
  - Xapian::ValueCountMatchSpy, 164
- get\_wdf
  - Xapian::PostingIterator, 103
  - Xapian::TermIterator, 152
- get\_weight
  - Xapian::ESetIterator, 72
  - Xapian::MSetIterator, 90
- has\_positions
  - Xapian::Database, 40
- include/xapian.h, 187
- include/xapian/database.h, 189
- include/xapian/dbfactory.h, 191
- include/xapian/document.h, 193
- include/xapian/enquire.h, 194
- include/xapian/errorhandler.h, 196
- include/xapian/expanddecider.h, 197
- include/xapian/matchspy.h, 198
- include/xapian/positioniterator.h, 200
- include/xapian/postingiterator.h, 201
- include/xapian/query.h, 202
- include/xapian/queryparser.h, 203
- include/xapian/sorter.h, 205
- include/xapian/stem.h, 206
- include/xapian/termgenerator.h, 207
- include/xapian/termiterator.h, 208
- include/xapian/types.h, 209

- include/xapian/unicode.h, 211
- include/xapian/valueiterator.h, 213
- increase\_termpos
  - Xapian::TermGenerator, 149
- index\_text
  - Xapian::TermGenerator, 148
- index\_text\_without\_positions
  - Xapian::TermGenerator, 148, 149
- Internal
  - Xapian::Query, 112
- iterator\_category
  - Xapian::ESetIterator, 71
  - Xapian::MSetIterator, 88
  - Xapian::PostingIterator, 102
  - Xapian::TermIterator, 151
  - Xapian::Utf8Iterator, 159
  - Xapian::ValueIterator, 168
- keep\_alive
  - Xapian::Database, 41
- left
  - Xapian::Utf8Iterator, 160
- major\_version
  - Xapian, 18
- MatchAll
  - Xapian::Query, 114
- MatchNothing
  - Xapian::Query, 114
- max\_size
  - Xapian::ESet, 68
  - Xapian::MSet, 85
- metadata\_keys\_begin
  - Xapian::Database, 44
- metadata\_keys\_end
  - Xapian::Database, 44
- minor\_version
  - Xapian, 18
- MSet
  - Xapian::MSet, 82
- MSetIterator
  - Xapian::MSetIterator, 89
- multivalues
  - Xapian::ValueCountMatchSpy, 166
- name
  - Xapian::BM25Weight, 25
  - Xapian::BoolWeight, 29
  - Xapian::TradWeight, 156
- Xapian::Weight, 173
- NumberValueRangeProcessor
  - Xapian::NumberValueRangeProcessor, 97
- op
  - Xapian::Query, 108
- OP\_AND
  - Xapian::Query, 108
- OP\_AND\_MAYBE
  - Xapian::Query, 109
- OP\_AND\_NOT
  - Xapian::Query, 108
- OP\_ELITE\_SET
  - Xapian::Query, 109
- OP\_FILTER
  - Xapian::Query, 109
- OP\_NEAR
  - Xapian::Query, 109
- OP\_OR
  - Xapian::Query, 108
- OP\_PHRASE
  - Xapian::Query, 109
- OP\_SCALE\_WEIGHT
  - Xapian::Query, 109
- op\_t
  - Xapian::Query, 108
- OP\_VALUE\_GE
  - Xapian::Query, 109
- OP\_VALUE\_LE
  - Xapian::Query, 109
- OP\_VALUE\_RANGE
  - Xapian::Query, 109
- OP\_XOR
  - Xapian::Query, 108
- operator \*
  - Xapian::ESetIterator, 72
  - Xapian::MSetIterator, 89
  - Xapian::PostingIterator, 103
  - Xapian::StringListUnserialiser, 137
  - Xapian::TermIterator, 152
  - Xapian::Utf8Iterator, 161
  - Xapian::ValueIterator, 169
- operator!=
  - Xapian, 19
  - Xapian::Utf8Iterator, 161
- operator()
  - Xapian::DateValueRangeProcessor, 46
  - Xapian::ErrorHandler, 65

- Xapian::ExpandDecider, 73
- Xapian::ExpandDeciderAnd, 75
- Xapian::ExpandDeciderFilterTerms, 77
- Xapian::MatchDecider, 78
- Xapian::MultipleMatchDecider, 92
- Xapian::MultiValueSorter, 94
- Xapian::NumberValueRangeProcessor, 97
- Xapian::SimpleStopper, 126
- Xapian::Sorter, 127
- Xapian::Stem, 130
- Xapian::Stopper, 131
- Xapian::StringValueRangeProcessor, 140
- Xapian::TermCountMatchSpy, 143
- Xapian::ValueCountMatchSpy, 165
- Xapian::ValueRangeProcessor, 170
- operator++
  - Xapian::ESetIterator, 71, 72
  - Xapian::MSetIterator, 89
  - Xapian::StringListUnserialiser, 138
  - Xapian::Utf8Iterator, 161
  - Xapian::ValueIterator, 168, 169
- operator–
  - Xapian::ESetIterator, 72
  - Xapian::MSetIterator, 89
- operator->
  - Xapian::ValueIterator, 169
- operator=
  - Xapian::Database, 39
  - Xapian::Document, 49
  - Xapian::Enquire, 56
  - Xapian::ESet, 68
  - Xapian::ESetIterator, 71
  - Xapian::MSet, 82
  - Xapian::MSetIterator, 89
  - Xapian::PositionIterator, 100
  - Xapian::PostingIterator, 103
  - Xapian::Query, 111, 112
  - Xapian::QueryParser, 118
  - Xapian::RSet, 123
  - Xapian::Stem, 130
  - Xapian::StringListUnserialiser, 137
  - Xapian::TermGenerator, 147
  - Xapian::TermIterator, 152
  - Xapian::ValueIterator, 168
  - Xapian::WritableDatabase, 179
- operator==
  - Xapian, 19
  - Xapian::PositionIterator, 100
  - Xapian::PostingIterator, 104
  - Xapian::StringListUnserialiser, 138
  - Xapian::Utf8Iterator, 161
- operator[]
  - Xapian::ESet, 69
  - Xapian::MSet, 86
- parse\_query
  - Xapian::QueryParser, 119
- percent
  - Xapian, 17
- PositionIterator
  - Xapian::PositionIterator, 100
- positionlist\_begin
  - Xapian::Database, 40
  - Xapian::PostingIterator, 103
  - Xapian::TermIterator, 152
- positionlist\_count
  - Xapian::TermIterator, 152
- positionlist\_end
  - Xapian::Database, 40
  - Xapian::PostingIterator, 103
  - Xapian::TermIterator, 152
- PostingIterator
  - Xapian::PostingIterator, 102
- postlist\_begin
  - Xapian::Database, 39
- postlist\_end
  - Xapian::Database, 39
- Query
  - Xapian::Query, 109–111
- QueryParser
  - Xapian::QueryParser, 117
- raw
  - Xapian::Utf8Iterator, 160
- remove\_document
  - Xapian::RSet, 124
- remove\_posting
  - Xapian::Document, 50
- remove\_spelling
  - Xapian::WritableDatabase, 184
- remove\_synonym
  - Xapian::WritableDatabase, 184
- remove\_term
  - Xapian::Document, 51
- remove\_value
  - Xapian::Document, 49

- reopen
  - Xapian::Database, 39
- replace\_document
  - Xapian::WritableDatabase, 182, 183
- revision
  - Xapian, 19
- RSet
  - Xapian::RSet, 123
- score\_categorisation
  - Xapian::CategorySelectMatchSpy, 33
- serialise
  - Xapian::BM25Weight, 25
  - Xapian::BoolWeight, 29
  - Xapian::Query, 113
  - Xapian::TradWeight, 156
  - Xapian::Weight, 173
- set\_collapse\_key
  - Xapian::Enquire, 57
- set\_cutoff
  - Xapian::Enquire, 58
- set\_data
  - Xapian::Document, 50
- set\_database
  - Xapian::QueryParser, 119
  - Xapian::TermGenerator, 147
- set\_default\_op
  - Xapian::QueryParser, 119
- set\_docid\_order
  - Xapian::Enquire, 57
- set\_document
  - Xapian::TermGenerator, 147
- set\_flags
  - Xapian::TermGenerator, 148
- set\_metadata
  - Xapian::WritableDatabase, 185
- set\_query
  - Xapian::Enquire, 56
- set\_sort\_by\_key
  - Xapian::Enquire, 59
- set\_sort\_by\_key\_then\_relevance
  - Xapian::Enquire, 59
- set\_sort\_by\_relevance
  - Xapian::Enquire, 58
- set\_sort\_by\_relevance\_then\_key
  - Xapian::Enquire, 60
- set\_sort\_by\_relevance\_then\_value
  - Xapian::Enquire, 60
- set\_sort\_by\_value
  - Xapian::Enquire, 58
- set\_sort\_by\_value\_then\_relevance
  - Xapian::Enquire, 59
- set\_stemmer
  - Xapian::QueryParser, 118
  - Xapian::TermGenerator, 147
- set\_stemming\_strategy
  - Xapian::QueryParser, 118
- set\_stopper
  - Xapian::QueryParser, 118
  - Xapian::TermGenerator, 147
- set\_termpos
  - Xapian::TermGenerator, 149
- set\_weighting\_scheme
  - Xapian::Enquire, 57
- SimpleStopper
  - Xapian::SimpleStopper, 126
- size
  - Xapian::ESet, 68
  - Xapian::MSet, 85
  - Xapian::RSet, 123
- skip\_to
  - Xapian::PostingIterator, 103
  - Xapian::TermIterator, 152
- sortable\_serialise
  - Xapian, 19
- sortable\_unserialise
  - Xapian, 20
- spellings\_begin
  - Xapian::Database, 42
- spellings\_end
  - Xapian::Database, 42
- Stem
  - Xapian::Stem, 128, 129
- stoplist\_begin
  - Xapian::QueryParser, 121
- StringListSerialiser
  - Xapian::StringListSerialiser, 134
- StringListUnserialiser
  - Xapian::StringListUnserialiser, 137
- StringValueRangeProcessor
  - Xapian::StringValueRangeProcessor, 139
- subquery\_list
  - Xapian::Query, 108
- swap
  - Xapian::ESet, 69
  - Xapian::MSet, 85
- synonym\_keys\_begin
  - Xapian::Database, 43

- synonym\_keys\_end
  - Xapian::Database, [43](#)
- synonyms\_begin
  - Xapian::Database, [42](#)
- synonyms\_end
  - Xapian::Database, [43](#)
- term\_exists
  - Xapian::Database, [41](#)
- termcount
  - Xapian, [17](#)
- termcount\_diff
  - Xapian, [17](#)
- TermCountMatchSpy
  - Xapian::TermCountMatchSpy, [142](#)
- TermGenerator
  - Xapian::TermGenerator, [147](#)
- TermIterator
  - Xapian::TermIterator, [151](#)
- termlist\_begin
  - Xapian::Database, [39](#)
  - Xapian::Document, [51](#)
- termlist\_count
  - Xapian::Document, [51](#)
- termlist\_end
  - Xapian::Database, [39](#)
  - Xapian::Document, [51](#)
- termpos
  - Xapian, [17](#)
- termpos\_diff
  - Xapian, [18](#)
- terms
  - Xapian::TermCountMatchSpy, [144](#)
- terms\_seen
  - Xapian::TermCountMatchSpy, [144](#)
- timeout
  - Xapian, [18](#)
- total
  - Xapian::ValueCountMatchSpy, [165](#)
- TradWeight
  - Xapian::TradWeight, [155](#)
- unserialise
  - Xapian::BM25Weight, [25](#)
  - Xapian::BoolWeight, [29](#)
  - Xapian::TradWeight, [156](#)
  - Xapian::Weight, [174](#)
- unstem\_begin
  - Xapian::QueryParser, [121](#)
- Utf8Iterator
  - Xapian::Utf8Iterator, [159](#), [160](#)
- value\_type
  - Xapian::MSet, [82](#)
- ValueCountMatchSpy
  - Xapian::ValueCountMatchSpy, [164](#)
- ValueIterator
  - Xapian::ValueIterator, [168](#)
- valueno
  - Xapian, [18](#)
- valueno\_diff
  - Xapian, [18](#)
- values
  - Xapian::ValueCountMatchSpy, [165](#)
- values\_begin
  - Xapian::Document, [51](#)
- values\_count
  - Xapian::Document, [51](#)
- values\_end
  - Xapian::Document, [51](#)
- version\_string
  - Xapian, [20](#)
- weight
  - Xapian, [18](#)
- WritableDatabase
  - Xapian::WritableDatabase, [178](#)
- Xapian, [11](#)
  - BAD\_VALUENO, [21](#)
  - DB\_CREATE, [21](#)
  - DB\_CREATE\_OR\_OPEN, [21](#)
  - DB\_CREATE\_OR\_OVERWRITE, [21](#)
  - DB\_OPEN, [21](#)
  - doccount, [17](#)
  - doccount\_diff, [17](#)
  - docid, [17](#)
  - doclength, [17](#)
  - major\_version, [18](#)
  - minor\_version, [18](#)
  - operator!=, [19](#)
  - operator==, [19](#)
  - percent, [17](#)
  - revision, [19](#)
  - sortable\_serialise, [19](#)
  - sortable\_unserialise, [20](#)
  - termcount, [17](#)
  - termcount\_diff, [17](#)
  - termpos, [17](#)

- termpos\_diff, 18
- timeout, 18
- valueno, 18
- valueno\_diff, 18
- version\_string, 20
- weight, 18
- XAPIAN\_DEPRECATED, 20
- Xapian::BM25Weight, 23
  - BM25Weight, 25
  - clone, 25
  - get\_maxextra, 26
  - get\_maxpart, 26
  - get\_sumextra, 26
  - get\_sumpart, 26
  - get\_sumpart\_needs\_doclength, 27
  - name, 25
  - serialise, 25
  - unserialise, 25
- Xapian::BoolWeight, 28
  - clone, 29
  - get\_maxextra, 30
  - get\_maxpart, 30
  - get\_sumextra, 30
  - get\_sumpart, 29
  - get\_sumpart\_needs\_doclength, 30
  - name, 29
  - serialise, 29
  - unserialise, 29
- Xapian::CategorySelectMatchSpy, 32
  - build\_numeric\_ranges, 33
  - CategorySelectMatchSpy, 33
  - score\_categorisation, 33
- Xapian::Database, 35
  - ~Database, 38
  - add\_database, 38
  - allterms\_begin, 40
  - allterms\_end, 40
  - Database, 38
  - get\_avlength, 41
  - get\_collection\_freq, 41
  - get\_description, 39
  - get\_doccount, 40
  - get\_doclength, 41
  - get\_document, 41
  - get\_lastdocid, 41
  - get\_metadata, 43
  - get\_spelling\_suggestion, 42
  - get\_termfreq, 41
  - has\_positions, 40
  - keep\_alive, 41
  - metadata\_keys\_begin, 44
  - metadata\_keys\_end, 44
  - operator=, 39
  - positionlist\_begin, 40
  - positionlist\_end, 40
  - postlist\_begin, 39
  - postlist\_end, 39
  - reopen, 39
  - spellings\_begin, 42
  - spellings\_end, 42
  - synonym\_keys\_begin, 43
  - synonym\_keys\_end, 43
  - synonyms\_begin, 42
  - synonyms\_end, 43
  - term\_exists, 41
  - termlist\_begin, 39
  - termlist\_end, 39
- Xapian::DateValueRangeProcessor, 45
  - DateValueRangeProcessor, 45
  - operator(), 46
- Xapian::Document, 47
  - ~Document, 49
  - add\_posting, 50
  - add\_term, 50
  - add\_value, 49
  - clear\_terms, 51
  - clear\_values, 49
  - Document, 48
  - get\_data, 49
  - get\_description, 52
  - get\_docid, 52
  - get\_value, 49
  - operator=, 49
  - remove\_posting, 50
  - remove\_term, 51
  - remove\_value, 49
  - set\_data, 50
  - termlist\_begin, 51
  - termlist\_count, 51
  - termlist\_end, 51
  - values\_begin, 51
  - values\_count, 51
  - values\_end, 51
- Xapian::Enquire, 53
  - ~Enquire, 56
  - Enquire, 56
  - get\_description, 64
  - get\_eset, 61, 62
  - get\_matching\_terms\_begin, 62, 63
  - get\_matching\_terms\_end, 63

- get\_mset, 60
- get\_query, 57
- operator=, 56
- set\_collapse\_key, 57
- set\_cutoff, 58
- set\_docid\_order, 57
- set\_query, 56
- set\_sort\_by\_key, 59
- set\_sort\_by\_key\_then\_relevance, 59
- set\_sort\_by\_relevance, 58
- set\_sort\_by\_relevance\_then\_key, 60
- set\_sort\_by\_relevance\_then\_value, 60
- set\_sort\_by\_value, 58
- set\_sort\_by\_value\_then\_relevance, 59
- set\_weighting\_scheme, 57
- XAPIAN\_DEPRECATED, 61, 64
- Xapian::ErrorHandler, 65
  - ~ErrorHandler, 65
  - ErrorHandler, 65
  - operator(), 65
- Xapian::ESet, 67
  - ~ESet, 68
  - back, 69
  - begin, 69
  - empty, 69
  - end, 69
  - ESet, 68
  - get\_description, 69
  - get\_ebound, 68
  - max\_size, 68
  - operator=, 68
  - operator[], 69
  - size, 68
  - swap, 69
- Xapian::ESetIterator, 70
  - ESetIterator, 71
  - get\_description, 72
  - get\_weight, 72
  - iterator\_category, 71
  - operator \*, 72
  - operator++, 71, 72
  - operator--, 72
  - operator=, 71
- Xapian::ExpandDecider, 73
  - ~ExpandDecider, 73
  - operator(), 73
- Xapian::ExpandDeciderAnd, 74
  - ExpandDeciderAnd, 75
- operator(), 75
- Xapian::ExpandDeciderFilterTerms, 76
  - ExpandDeciderFilterTerms, 76
  - operator(), 77
- Xapian::MatchDecider, 78
  - ~MatchDecider, 78
  - operator(), 78
- Xapian::MSet, 80
  - ~MSet, 82
  - back, 85
  - begin, 85
  - convert\_to\_percent, 83
  - empty, 85
  - end, 85
  - fetch, 82, 83
  - get\_description, 86
  - get\_firstitem, 84
  - get\_matches\_estimated, 84
  - get\_matches\_lower\_bound, 84
  - get\_matches\_upper\_bound, 84
  - get\_max\_attained, 85
  - get\_max\_possible, 85
  - get\_termfreq, 83
  - get\_termweight, 84
  - max\_size, 85
  - MSet, 82
  - operator=, 82
  - operator[], 86
  - size, 85
  - swap, 85
  - value\_type, 82
- Xapian::MSetIterator, 87
  - get\_collapse\_count, 90
  - get\_collapse\_key, 90
  - get\_description, 91
  - get\_document, 89
  - get\_percent, 90
  - get\_rank, 90
  - get\_weight, 90
  - iterator\_category, 88
  - MSetIterator, 89
  - operator \*, 89
  - operator++, 89
  - operator--, 89
  - operator=, 89
- Xapian::MultipleMatchDecider, 92
  - append, 92
  - operator(), 92
- Xapian::MultiValueSorter, 94
  - operator(), 94

- Xapian::NumberValueRangeProcessor, 96
  - NumberValueRangeProcessor, 97
  - operator(), 97
- Xapian::PositionIterator, 99
  - ~PositionIterator, 100
  - get\_description, 100
  - operator=, 100
  - operator==, 100
  - PositionIterator, 100
- Xapian::PostingIterator, 101
  - ~PostingIterator, 102
  - get\_description, 103
  - get\_doclength, 103
  - get\_wdf, 103
  - iterator\_category, 102
  - operator \*, 103
  - operator=, 103
  - operator==, 104
  - positionlist\_begin, 103
  - positionlist\_end, 103
  - PostingIterator, 102
  - skip\_to, 103
- Xapian::Query, 105
  - ~Internal, 111
  - ~Query, 109
  - add\_subquery, 113
  - empty, 112
  - end\_construction, 113
  - get\_description, 112, 113
  - get\_length, 111, 113
  - get\_parameter, 113
  - get\_terms, 113
  - get\_terms\_begin, 111
  - get\_terms\_end, 112
  - Internal, 112
  - MatchAll, 114
  - MatchNothing, 114
  - op, 108
  - OP\_AND, 108
  - OP\_AND\_MAYBE, 109
  - OP\_AND\_NOT, 108
  - OP\_ELITE\_SET, 109
  - OP\_FILTER, 109
  - OP\_NEAR, 109
  - OP\_OR, 108
  - OP\_PHRASE, 109
  - OP\_SCALE\_WEIGHT, 109
  - op\_t, 108
  - OP\_VALUE\_GE, 109
  - OP\_VALUE\_LE, 109
  - OP\_VALUE\_RANGE, 109
  - OP\_XOR, 108
  - operator=, 111, 112
  - Query, 109–111
  - serialise, 113
  - subquery\_list, 108
  - XAPIAN\_DEPRECATED, 111
- Xapian::QueryParser, 115
  - ~QueryParser, 118
  - add\_boolean\_prefix, 120
  - add\_prefix, 119
  - add\_valuerangeprocessor, 121
  - feature\_flag, 116
  - FLAG\_AUTO\_MULTIWORD\_-SYNONYMS, 117
  - FLAG\_AUTO\_SYNONYMS, 117
  - FLAG\_BOOLEAN, 117
  - FLAG\_BOOLEAN\_ANY\_CASE, 117
  - FLAG\_LOVEHATE, 117
  - FLAG\_PARTIAL, 117
  - FLAG\_PHRASE, 117
  - FLAG\_PURE\_NOT, 117
  - FLAG\_SPELLING\_-CORRECTION, 117
  - FLAG\_SYNONYM, 117
  - FLAG\_WILDCARD, 117
  - get\_corrected\_query\_string, 121
  - get\_default\_op, 119
  - get\_description, 121
  - operator=, 118
  - parse\_query, 119
  - QueryParser, 117
  - set\_database, 119
  - set\_default\_op, 119
  - set\_stemmer, 118
  - set\_stemming\_strategy, 118
  - set\_stopper, 118
  - stoplist\_begin, 121
  - unstem\_begin, 121
- Xapian::RSet, 122
  - ~RSet, 123
  - add\_document, 123
  - contains, 124
  - empty, 123
  - get\_description, 124
  - operator=, 123
  - remove\_document, 124
  - RSet, 123



- size, 123
- Xapian::SimpleStopper, 125
  - ~SimpleStopper, 126
  - add, 126
  - get\_description, 126
  - operator(), 126
  - SimpleStopper, 126
- Xapian::Sorter, 127
  - ~Sorter, 127
  - operator(), 127
- Xapian::Stem, 128
  - ~Stem, 129
  - get\_available\_languages, 130
  - get\_description, 130
  - operator(), 130
  - operator=, 130
  - Stem, 128, 129
- Xapian::Stopper, 131
  - ~Stopper, 131
  - get\_description, 131
  - operator(), 131
- Xapian::StringAndFrequency, 133
- Xapian::StringListSerialiser, 134
  - append, 135
  - get, 135
  - StringListSerialiser, 134
- Xapian::StringListUnserialiser, 136
  - ~StringListUnserialiser, 137
  - operator \*, 137
  - operator++, 138
  - operator=, 137
  - operator==, 138
  - StringListUnserialiser, 137
- Xapian::StringValueRangeProcessor, 139
  - operator(), 140
  - StringValueRangeProcessor, 139
- Xapian::TermCountMatchSpy, 141
  - add\_prefix, 142
  - documents\_seen, 144
  - get\_documents\_seen, 143
  - get\_terms, 142
  - get\_terms\_seen, 143
  - get\_top\_terms, 143
  - operator(), 143
  - TermCountMatchSpy, 142
  - terms, 144
  - terms\_seen, 144
- Xapian::TermGenerator, 145
  - ~TermGenerator, 147
  - FLAG\_SPELLING, 146
  - flags, 146
  - get\_description, 149
  - get\_document, 147
  - get\_termpos, 149
  - increase\_termpos, 149
  - index\_text, 148
  - index\_text\_without\_positions, 148, 149
  - operator=, 147
  - set\_database, 147
  - set\_document, 147
  - set\_flags, 148
  - set\_stemmer, 147
  - set\_stopper, 147
  - set\_termpos, 149
  - TermGenerator, 147
- Xapian::TermIterator, 150
  - ~TermIterator, 151
  - get\_description, 152
  - get\_termfreq, 152
  - get\_wdf, 152
  - iterator\_category, 151
  - operator \*, 152
  - operator=, 152
  - positionlist\_begin, 152
  - positionlist\_count, 152
  - positionlist\_end, 152
  - skip\_to, 152
  - TermIterator, 151
- Xapian::TradWeight, 154
  - clone, 156
  - get\_maxextra, 157
  - get\_maxpart, 156
  - get\_sumextra, 157
  - get\_sumpart, 156
  - get\_sumpart\_needs\_doclength, 157
  - name, 156
  - serialise, 156
  - TradWeight, 155
  - unserialise, 156
- Xapian::Utf8Iterator, 158
  - assign, 160
  - iterator\_category, 159
  - left, 160
  - operator \*, 161
  - operator!=, 161
  - operator++, 161
  - operator==, 161
  - raw, 160
  - Utf8Iterator, 159, 160

- Xapian::ValueCountMatchSpy, 163
  - add\_slot, 164
  - get\_top\_values, 165
  - get\_total, 165
  - get\_values, 164
  - multivalues, 166
  - operator(), 165
  - total, 165
  - ValueCountMatchSpy, 164
  - values, 165
- Xapian::ValueIterator, 167
  - get\_description, 169
  - get\_valueno, 169
  - iterator\_category, 168
  - operator \*, 169
  - operator++, 168, 169
  - operator->, 169
  - operator=, 168
  - ValueIterator, 168
- Xapian::ValueRangeProcessor, 170
  - ~ValueRangeProcessor, 170
  - operator(), 170
- Xapian::Weight, 172
  - create, 173
  - get\_maxextra, 175
  - get\_maxpart, 174
  - get\_sumextra, 174
  - get\_sumpart, 174
  - get\_sumpart\_needs\_doclength, 175
  - name, 173
  - serialise, 173
  - unserialise, 174
- Xapian::WritableDatabase, 176
  - ~WritableDatabase, 178
  - add\_document, 181
  - add\_spelling, 184
  - add\_synonym, 184
  - begin\_transaction, 179
  - cancel\_transaction, 181
  - clear\_synonyms, 185
  - commit\_transaction, 180
  - delete\_document, 182
  - flush, 179
  - get\_description, 186
  - operator=, 179
  - remove\_spelling, 184
  - remove\_synonym, 184
  - replace\_document, 182, 183
  - set\_metadata, 185
  - WritableDatabase, 178
- XAPIAN\_DEPRECATED
  - Xapian, 20
  - Xapian::Enquire, 61, 64
  - Xapian::Query, 111