Contents

1 Deprecated List 1

2 Namespace Index 3
   2.1 Namespace List .............................................. 3

3 Class Index 5
   3.1 Class Hierarchy ............................................... 5

4 Class Index 9
   4.1 Class List .................................................. 9

5 File Index 13
   5.1 File List .................................................. 13

6 Namespace Documentation 15
   6.1 Xapian Namespace Reference ................................. 15
      6.1.1 Detailed Description .................................. 21
      6.1.2 Typedef Documentation ................................ 22
         6.1.2.1 doccount ............................................. 22
         6.1.2.2 doccount_diff ....................................... 22
         6.1.2.3 docid ................................................ 22
         6.1.2.4 doclength ............................................ 22
         6.1.2.5 percent .............................................. 22
         6.1.2.6 termcount ............................................. 22
         6.1.2.7 termcount_diff ...................................... 22
         6.1.2.8 termpos_diff ......................................... 23
         6.1.2.9 timeout ............................................... 23
7.7.2 Constructor & Destructor Documentation ........................................ 60
  7.7.2.1 DatabaseCreateError .................................................. 60
  7.7.2.2 DatabaseCreateError .................................................. 61

7.8 Xapian::DatabaseError Class Reference ........................................... 61
  7.8.1 Detailed Description ....................................................... 62
  7.8.2 Constructor & Destructor Documentation .................................... 62
     7.8.2.1 DatabaseError ....................................................... 62
     7.8.2.2 DatabaseError ....................................................... 62

7.9 Xapian::DatabaseLockError Class Reference ...................................... 62
  7.9.1 Detailed Description ...................................................... 63
  7.9.2 Constructor & Destructor Documentation .................................... 63
     7.9.2.1 DatabaseLockError .................................................. 63
     7.9.2.2 DatabaseLockError .................................................. 64

7.10 Xapian::DatabaseModifiedError Class Reference .................................. 64
  7.10.1 Detailed Description .................................................... 65
  7.10.2 Constructor & Destructor Documentation .................................... 65
     7.10.2.1 DatabaseModifiedError ............................................. 66
     7.10.2.2 DatabaseModifiedError ............................................. 66

7.11 Xapian::DatabaseOpeningError Class Reference .................................. 66
  7.11.1 Detailed Description .................................................... 67
  7.11.2 Constructor & Destructor Documentation .................................... 67
     7.11.2.1 DatabaseOpeningError .............................................. 68
     7.11.2.2 DatabaseOpeningError .............................................. 68

7.12 Xapian::DatabaseVersionError Class Reference .................................. 68
  7.12.1 Detailed Description .................................................... 69
  7.12.2 Constructor & Destructor Documentation .................................... 70
     7.12.2.1 DatabaseVersionError .............................................. 70
     7.12.2.2 DatabaseVersionError .............................................. 70

7.13 Xapian::DateValueRangeProcessor Class Reference .............................. 70
  7.13.1 Detailed Description .................................................... 71
  7.13.2 Constructor & Destructor Documentation .................................... 71
     7.13.2.1 DateValueRangeProcessor ......................................... 72
     7.13.2.2 DateValueRangeProcessor ......................................... 72
     7.13.2.3 DateValueRangeProcessor ......................................... 72
7.13.3 Member Function Documentation . . . . . . . . . . . . . . . . 73
  7.13.3.1 operator() . . . . . . . . . . . . . . . . . . . . . . . 73

7.14 Xapian::DecreasingValueWeightPostingSource Class Reference . . . . 74
  7.14.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . 75
  7.14.2 Member Function Documentation . . . . . . . . . . . . . . . . 76
    7.14.2.1 check . . . . . . . . . . . . . . . . . . . . . . . . . . 76
    7.14.2.2 clone . . . . . . . . . . . . . . . . . . . . . . . . . . 76
    7.14.2.3 get_description . . . . . . . . . . . . . . . . . . . . . 77
    7.14.2.4 get_weight . . . . . . . . . . . . . . . . . . . . . . . 77
    7.14.2.5 init . . . . . . . . . . . . . . . . . . . . . . . . . . 77
    7.14.2.6 name . . . . . . . . . . . . . . . . . . . . . . . . . . 78
    7.14.2.7 next . . . . . . . . . . . . . . . . . . . . . . . . . . 78
    7.14.2.8 serialise . . . . . . . . . . . . . . . . . . . . . . . . 79
    7.14.2.9 skip_to . . . . . . . . . . . . . . . . . . . . . . . . . 79
    7.14.2.10 unserialise . . . . . . . . . . . . . . . . . . . . . . . 79

7.15 Xapian::DocNotFoundError Class Reference . . . . . . . . . . . . . . . . 80
  7.15.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . 81
  7.15.2 Constructor & Destructor Documentation . . . . . . . . . . . . 81
    7.15.2.1 DocNotFoundError . . . . . . . . . . . . . . . . . . . . 81
    7.15.2.2 DocNotFoundError . . . . . . . . . . . . . . . . . . . . 81

7.16 Xapian::Document Class Reference . . . . . . . . . . . . . . . . . . . 81
  7.16.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . 83
  7.16.2 Constructor & Destructor Documentation . . . . . . . . . . . . 83
    7.16.2.1 Document . . . . . . . . . . . . . . . . . . . . . . . . 83
  7.16.3 Member Function Documentation . . . . . . . . . . . . . . . . . 83
    7.16.3.1 add_boolean_term . . . . . . . . . . . . . . . . . . . . 83
    7.16.3.2 add_posting . . . . . . . . . . . . . . . . . . . . . . . 84
    7.16.3.3 add_term . . . . . . . . . . . . . . . . . . . . . . . . 84
    7.16.3.4 add_value . . . . . . . . . . . . . . . . . . . . . . . . 84
    7.16.3.5 get_data . . . . . . . . . . . . . . . . . . . . . . . . . 85
    7.16.3.6 get_docid . . . . . . . . . . . . . . . . . . . . . . . . 85
    7.16.3.7 get_value . . . . . . . . . . . . . . . . . . . . . . . . 85
    7.16.3.8 operator= . . . . . . . . . . . . . . . . . . . . . . . . 85
    7.16.3.9 remove_posting . . . . . . . . . . . . . . . . . . . . . 85
7.25.2 Constructor & Destructor Documentation . . . . . . . . . . . . . 109
7.25.2.1 ExpandDeciderFilterTerms . . . . . . . . . . . . . . . 109
7.25.3 Member Function Documentation . . . . . . . . . . . . . . . 109
7.25.3.1 operator() . . . . . . . . . . . . . . . . . . . . . . . 109
7.26 Xapian::FeatureUnavailableError Class Reference . . . . . . . . . . . 109
7.26.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . 110
7.26.2 Constructor & Destructor Documentation . . . . . . . . . . . . . 110
7.26.2.1 FeatureUnavailableError . . . . . . . . . . . . . . . 110
7.26.2.2 FeatureUnavailableError . . . . . . . . . . . . . . . 111
7.27 Xapian::FieldProcessor Struct Reference . . . . . . . . . . . . . . . . 111
7.27.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . 111
7.27.2 Member Function Documentation . . . . . . . . . . . . . . . 111
7.27.2.1 operator() . . . . . . . . . . . . . . . . . . . . . . . 111
7.28 Xapian::FixedWeightPostingSource Class Reference . . . . . . . . . . 112
7.28.1 Detailed Description . . . . . . . . . . . . . . . . . . . . . . 113
7.28.2 Constructor & Destructor Documentation . . . . . . . . . . . . . 113
7.28.2.1 FixedWeightPostingSource . . . . . . . . . . . . . . 113
7.28.3 Member Function Documentation . . . . . . . . . . . . . . . 113
7.28.3.1 at_end . . . . . . . . . . . . . . . . . . . . . . . . . 114
7.28.3.2 check . . . . . . . . . . . . . . . . . . . . . . . . . . 114
7.28.3.3 clone . . . . . . . . . . . . . . . . . . . . . . . . . . 114
7.28.3.4 get_description . . . . . . . . . . . . . . . . . . . . . 115
7.28.3.5 get_docid . . . . . . . . . . . . . . . . . . . . . . . . 115
7.28.3.6 get_termfreq_est . . . . . . . . . . . . . . . . . . . . 115
7.28.3.7 get_termfreq_max . . . . . . . . . . . . . . . . . . . 116
7.28.3.8 get_termfreq_min . . . . . . . . . . . . . . . . . . . 116
7.28.3.9 get_weight . . . . . . . . . . . . . . . . . . . . . . . 116
7.28.3.10 init . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
7.28.3.11 name . . . . . . . . . . . . . . . . . . . . . . . . . . 117
7.28.3.12 next . . . . . . . . . . . . . . . . . . . . . . . . . . 117
7.28.3.13 serialise . . . . . . . . . . . . . . . . . . . . . . . . 117
7.28.3.14 skip_to . . . . . . . . . . . . . . . . . . . . . . . . . 118
7.28.3.15 unserialise . . . . . . . . . . . . . . . . . . . . . . . 118
7.29 Xapian::GreatCircleMetric Class Reference . . . . . . . . . . . . . . . 119
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.61.2.2</td>
<td>Stem</td>
</tr>
<tr>
<td>7.61.2.3</td>
<td>Stem</td>
</tr>
<tr>
<td>7.61.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>7.61.3.1</td>
<td>get_available_languages</td>
</tr>
<tr>
<td>7.61.3.2</td>
<td>operator()</td>
</tr>
<tr>
<td>7.62</td>
<td>Xapian::StemImplementation Struct Reference</td>
</tr>
<tr>
<td>7.62.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>7.63</td>
<td>Xapian::Stopper Class Reference</td>
</tr>
<tr>
<td>7.63.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>7.63.2</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>7.63.2.1</td>
<td>operator()</td>
</tr>
<tr>
<td>7.64</td>
<td>Xapian::StringValueRangeProcessor Class Reference</td>
</tr>
<tr>
<td>7.64.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>7.64.2</td>
<td>Constructor &amp; Destructor Documentation</td>
</tr>
<tr>
<td>7.64.2.1</td>
<td>StringValueRangeProcessor</td>
</tr>
<tr>
<td>7.64.2.2</td>
<td>StringValueRangeProcessor</td>
</tr>
<tr>
<td>7.64.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>7.64.3.1</td>
<td>operator()</td>
</tr>
<tr>
<td>7.65</td>
<td>Xapian::TermGenerator Class Reference</td>
</tr>
<tr>
<td>7.65.1</td>
<td>Detailed Description</td>
</tr>
<tr>
<td>7.65.2</td>
<td>Member Enumeration Documentation</td>
</tr>
<tr>
<td>7.65.2.1</td>
<td>flags</td>
</tr>
<tr>
<td>7.65.3</td>
<td>Member Function Documentation</td>
</tr>
<tr>
<td>7.65.3.1</td>
<td>increase_termpos</td>
</tr>
<tr>
<td>7.65.3.2</td>
<td>index_text</td>
</tr>
<tr>
<td>7.65.3.3</td>
<td>index_text</td>
</tr>
<tr>
<td>7.65.3.4</td>
<td>index_text_without_positions</td>
</tr>
<tr>
<td>7.65.3.5</td>
<td>index_text_without_positions</td>
</tr>
<tr>
<td>7.65.3.6</td>
<td>set_flags</td>
</tr>
<tr>
<td>7.65.3.7</td>
<td>set_max_word_length</td>
</tr>
<tr>
<td>7.65.3.8</td>
<td>set_stemming_strategy</td>
</tr>
<tr>
<td>7.65.3.9</td>
<td>set_stopper</td>
</tr>
<tr>
<td>7.65.3.10</td>
<td>set_termpos</td>
</tr>
<tr>
<td>7.66</td>
<td>Xapian::TermIterator Class Reference</td>
</tr>
</tbody>
</table>

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.73.9 skip_to</td>
<td>236</td>
</tr>
<tr>
<td>7.74.3.9 skip_to</td>
<td>236</td>
</tr>
<tr>
<td>7.74.4 Member Data Documentation</td>
<td>237</td>
</tr>
<tr>
<td>7.74.4.1 termfreq_est</td>
<td>237</td>
</tr>
<tr>
<td>7.74.4.2 termfreq_max</td>
<td>237</td>
</tr>
<tr>
<td>7.74.4.3 termfreq_min</td>
<td>237</td>
</tr>
<tr>
<td>7.75 Xapian::ValueRangeProcessor Struct Reference</td>
<td>237</td>
</tr>
<tr>
<td>7.75.1 Detailed Description</td>
<td>238</td>
</tr>
<tr>
<td>7.75.2 Member Function Documentation</td>
<td>238</td>
</tr>
<tr>
<td>7.75.2.1 operator()</td>
<td>238</td>
</tr>
<tr>
<td>7.76 Xapian::ValueSetMatchDecider Class Reference</td>
<td>239</td>
</tr>
<tr>
<td>7.76.1 Detailed Description</td>
<td>240</td>
</tr>
<tr>
<td>7.76.2 Constructor &amp; Destructor Documentation</td>
<td>240</td>
</tr>
<tr>
<td>7.76.2.1 ValueSetMatchDecider</td>
<td>240</td>
</tr>
<tr>
<td>7.76.3 Member Function Documentation</td>
<td>240</td>
</tr>
<tr>
<td>7.76.3.1 add_value</td>
<td>240</td>
</tr>
<tr>
<td>7.76.3.2 operator()</td>
<td>240</td>
</tr>
<tr>
<td>7.76.3.3 remove_value</td>
<td>241</td>
</tr>
<tr>
<td>7.77 Xapian::ValueWeightPostingSource Class Reference</td>
<td>241</td>
</tr>
<tr>
<td>7.77.1 Detailed Description</td>
<td>242</td>
</tr>
<tr>
<td>7.77.2 Constructor &amp; Destructor Documentation</td>
<td>242</td>
</tr>
<tr>
<td>7.77.2.1 ValueWeightPostingSource</td>
<td>242</td>
</tr>
<tr>
<td>7.77.3 Member Function Documentation</td>
<td>242</td>
</tr>
<tr>
<td>7.77.3.1 clone</td>
<td>243</td>
</tr>
<tr>
<td>7.77.3.2 get_description</td>
<td>243</td>
</tr>
<tr>
<td>7.77.3.3 get_weight</td>
<td>243</td>
</tr>
<tr>
<td>7.77.3.4 init</td>
<td>244</td>
</tr>
<tr>
<td>7.77.3.5 name</td>
<td>244</td>
</tr>
<tr>
<td>7.77.3.6 serialise</td>
<td>245</td>
</tr>
<tr>
<td>7.77.3.7 unserialise</td>
<td>245</td>
</tr>
<tr>
<td>7.78 Xapian::Weight Class Reference</td>
<td>245</td>
</tr>
<tr>
<td>7.78.1 Detailed Description</td>
<td>247</td>
</tr>
<tr>
<td>7.78.2 Constructor &amp; Destructor Documentation</td>
<td>247</td>
</tr>
<tr>
<td>7.78.2.1 ~Weight</td>
<td>247</td>
</tr>
<tr>
<td>7.78.2.2 Weight</td>
<td>247</td>
</tr>
</tbody>
</table>
### 7.79.3.14 remove_synonym
- Page 259

### 7.79.3.15 replace_document
- Page 260

### 7.79.3.16 replace_document
- Page 260

### 7.79.3.17 set_metadata
- Page 261

### 8 File Documentation
- Page 263

#### 8.1 xapian/error.h File Reference
- Page 263
  - 8.1.1 Detailed Description
- Page 264

#### 8.2 xapian/version.h File Reference
- Page 264
  - 8.2.1 Detailed Description
- Page 265
  - 8.2.2 Define Documentation
    - 8.2.2.1 XAPIAN_MAJOR_VERSION
    - 8.2.2.2 XAPIAN_MINOR_VERSION
    - 8.2.2.3 XAPIAN_REVISION

#### 8.3 xapian.h File Reference
- Page 265
  - 8.3.1 Detailed Description

#### 8.4 xapian/attributes.h File Reference
- Page 266
  - 8.4.1 Detailed Description
  - 8.4.2 Define Documentation
    - 8.4.2.1 XAPIAN_CONST_FUNCTION

#### 8.5 xapian/compactor.h File Reference
- Page 267
  - 8.5.1 Detailed Description

#### 8.6 xapian/database.h File Reference
- Page 267
  - 8.6.1 Detailed Description

#### 8.7 xapian/dbfactory.h File Reference
- Page 268
  - 8.7.1 Detailed Description

#### 8.8 xapian/document.h File Reference
- Page 270
  - 8.8.1 Detailed Description

#### 8.9 xapian/enquire.h File Reference
- Page 270
  - 8.9.1 Detailed Description

#### 8.10 xapian/errorhandler.h File Reference
- Page 271
  - 8.10.1 Detailed Description

#### 8.11 xapian/expanddecider.h File Reference
- Page 271
  - 8.11.1 Detailed Description

---

*Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen*
<table>
<thead>
<tr>
<th>Section</th>
<th>File Reference</th>
<th>Detailed Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.12</td>
<td>xapian/geospatial.h</td>
<td>272</td>
</tr>
<tr>
<td>8.12.1</td>
<td>Detailed Description</td>
<td>273</td>
</tr>
<tr>
<td>8.13</td>
<td>xapian/keymaker.h</td>
<td>273</td>
</tr>
<tr>
<td>8.13.1</td>
<td>Detailed Description</td>
<td>273</td>
</tr>
<tr>
<td>8.14</td>
<td>xapian/matchspy.h</td>
<td>274</td>
</tr>
<tr>
<td>8.14.1</td>
<td>Detailed Description</td>
<td>274</td>
</tr>
<tr>
<td>8.15</td>
<td>xapian/positioniterator.h</td>
<td>274</td>
</tr>
<tr>
<td>8.15.1</td>
<td>Detailed Description</td>
<td>275</td>
</tr>
<tr>
<td>8.16</td>
<td>xapian/postingiterator.h</td>
<td>275</td>
</tr>
<tr>
<td>8.16.1</td>
<td>Detailed Description</td>
<td>275</td>
</tr>
<tr>
<td>8.17</td>
<td>xapian/postingsource.h</td>
<td>275</td>
</tr>
<tr>
<td>8.17.1</td>
<td>Detailed Description</td>
<td>276</td>
</tr>
<tr>
<td>8.18</td>
<td>xapian/query.h</td>
<td>276</td>
</tr>
<tr>
<td>8.18.1</td>
<td>Detailed Description</td>
<td>276</td>
</tr>
<tr>
<td>8.19</td>
<td>xapian/queryparser.h</td>
<td>276</td>
</tr>
<tr>
<td>8.19.1</td>
<td>Detailed Description</td>
<td>277</td>
</tr>
<tr>
<td>8.20</td>
<td>xapian/registry.h</td>
<td>277</td>
</tr>
<tr>
<td>8.20.1</td>
<td>Detailed Description</td>
<td>278</td>
</tr>
<tr>
<td>8.21</td>
<td>xapian/stem.h</td>
<td>278</td>
</tr>
<tr>
<td>8.21.1</td>
<td>Detailed Description</td>
<td>278</td>
</tr>
<tr>
<td>8.22</td>
<td>xapian/termgenerator.h</td>
<td>278</td>
</tr>
<tr>
<td>8.22.1</td>
<td>Detailed Description</td>
<td>279</td>
</tr>
<tr>
<td>8.23</td>
<td>xapian/termiterator.h</td>
<td>279</td>
</tr>
<tr>
<td>8.23.1</td>
<td>Detailed Description</td>
<td>279</td>
</tr>
<tr>
<td>8.24</td>
<td>xapian/types.h</td>
<td>280</td>
</tr>
<tr>
<td>8.24.1</td>
<td>Detailed Description</td>
<td>281</td>
</tr>
<tr>
<td>8.25</td>
<td>xapian/unicode.h</td>
<td>281</td>
</tr>
<tr>
<td>8.25.1</td>
<td>Detailed Description</td>
<td>282</td>
</tr>
<tr>
<td>8.26</td>
<td>xapian/valueiterator.h</td>
<td>282</td>
</tr>
<tr>
<td>8.26.1</td>
<td>Detailed Description</td>
<td>282</td>
</tr>
<tr>
<td>8.27</td>
<td>xapian/valuesetmatchdecider.h</td>
<td>282</td>
</tr>
<tr>
<td>8.27.1</td>
<td>Detailed Description</td>
<td>283</td>
</tr>
<tr>
<td>8.28</td>
<td>xapian/weight.h</td>
<td>283</td>
</tr>
<tr>
<td>8.28.1</td>
<td>Detailed Description</td>
<td>283</td>
</tr>
</tbody>
</table>
Chapter 1

Deprecated List

**Member Xapian::percent**
This type is deprecated as of Xapian 1.3.0 - use the standard type int instead, which should work with older Xapian too.

**Member Xapian::timeout**
This type is deprecated as of Xapian 1.3.0 - use the standard POSIX type useconds_t instead, which should work with older Xapian too.

**Member Xapian::weight**
This type is deprecated as of Xapian 1.3.0 - use the standard C++ type double instead, which should work with older Xapian too.
Chapter 2

Namespace Index

2.1 Namespace List

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

- **Xapian**
  The Xapian namespace contains public interfaces for the Xapian library

- **Xapian::Auto**
  Database factory functions which determine the database type automatically

- **Xapian::Brass**
  Database factory functions for the brass backend

- **Xapian::Chert**
  Database factory functions for the chert backend

- **Xapian::InMemory**
  Database factory functions for the inmemory backend

- **Xapian::Remote**
  Database factory functions for the remote backend

- **Xapian::Unicode**
  Functions associated with handling Unicode characters
Chapter 3

Class Index

3.1 Class Hierarchy

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

Xapian::Compactor .......................... 43
Xapian::Database ................................ 46
Xapian::WritableDatabase ...................... 251
Xapian::Document ................................ 81
Xapian::Enquire ................................ 87
Xapian::Error ................................ 99
Xapian::LogicError ............................. 142
Xapian::AssertionError ......................... 35
Xapian::InvalidArgumentError ................. 123
Xapian::InvalidOperationError ................. 124
Xapian::UnimplementedError ................... 214
Xapian::RuntimeError .......................... 190
Xapian::DatabaseError .......................... 61
Xapian::DatabaseCorruptError ................... 58
Xapian::DatabaseCreateError ..................... 59
Xapian::DatabaseLockError ...................... 62
Xapian::DatabaseModifiedError .................. 64
Xapian::DatabaseOpeningError ................... 66
Xapian::DatabaseVersionError .................... 68
Xapian::DocNotFoundError ....................... 80
Xapian::FeatureUnavailableError ............... 109
Xapian::InternalError ........................... 121
Xapian::NetworkError ............................ 156
Xapian::NetworkTimeoutError ..................... 158
Xapian::QueryParserError ...................... 183
Xapian::RangeError ................................ 184
Xapian::SerialisationError ..................... 191
Xapian::ErrorHandler ........................... 100
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::ESet</td>
<td>101</td>
</tr>
<tr>
<td>Xapian::ESetIterator</td>
<td>102</td>
</tr>
<tr>
<td>Xapian::ExpandDecider</td>
<td>104</td>
</tr>
<tr>
<td>Xapian::ExpandDeciderAnd</td>
<td>105</td>
</tr>
<tr>
<td>Xapian::ExpandDeciderFilterPrefix</td>
<td>107</td>
</tr>
<tr>
<td>Xapian::ExpandDeciderFilterTerms</td>
<td>108</td>
</tr>
<tr>
<td>Xapian::FieldProcessor</td>
<td>111</td>
</tr>
<tr>
<td>Xapian::KeyMaker</td>
<td>126</td>
</tr>
<tr>
<td>Xapian::LatLongDistanceKeyMaker</td>
<td>131</td>
</tr>
<tr>
<td>Xapian::MultiValueKeyMaker</td>
<td>155</td>
</tr>
<tr>
<td>Xapian::LatLongCoord</td>
<td>127</td>
</tr>
<tr>
<td>Xapian::LatLongCoords</td>
<td>129</td>
</tr>
<tr>
<td>Xapian::LatLongCoordsIterator</td>
<td>131</td>
</tr>
<tr>
<td>Xapian::LatLongMetric</td>
<td>139</td>
</tr>
<tr>
<td>Xapian::GreatCircleMetric</td>
<td>119</td>
</tr>
<tr>
<td>Xapian::MatchDecider</td>
<td>143</td>
</tr>
<tr>
<td>Xapian::ValueSetMatchDecider</td>
<td>239</td>
</tr>
<tr>
<td>Xapian::MatchSpy</td>
<td>144</td>
</tr>
<tr>
<td>Xapian::ValueCountMatchSpy</td>
<td>220</td>
</tr>
<tr>
<td>Xapian::MSet</td>
<td>148</td>
</tr>
<tr>
<td>Xapian::MSetIterator</td>
<td>152</td>
</tr>
<tr>
<td>Xapian::PositionIterator</td>
<td>162</td>
</tr>
<tr>
<td>Xapian::PostingIterator</td>
<td>164</td>
</tr>
<tr>
<td>Xapian::PostingSource</td>
<td>165</td>
</tr>
<tr>
<td>Xapian::FixedWeightPostingSource</td>
<td>112</td>
</tr>
<tr>
<td>Xapian::ValuePostingSource</td>
<td>231</td>
</tr>
<tr>
<td>Xapian::LatLongDistancePostingSource</td>
<td>132</td>
</tr>
<tr>
<td>Xapian::ValueMapPostingSource</td>
<td>227</td>
</tr>
<tr>
<td>Xapian::ValueWeightPostingSource</td>
<td>241</td>
</tr>
<tr>
<td>Xapian::DecreasingValueWeightPostingSource</td>
<td>74</td>
</tr>
<tr>
<td>Xapian::Query</td>
<td>173</td>
</tr>
<tr>
<td>Xapian::QueryParser</td>
<td>175</td>
</tr>
<tr>
<td>Xapian::Registry</td>
<td>186</td>
</tr>
<tr>
<td>Xapian::RSet</td>
<td>189</td>
</tr>
<tr>
<td>Xapian::Stem</td>
<td>194</td>
</tr>
<tr>
<td>Xapian::StemImplementation</td>
<td>197</td>
</tr>
<tr>
<td>Xapian::Stopper</td>
<td>198</td>
</tr>
<tr>
<td>Xapian::SimpleStopper</td>
<td>193</td>
</tr>
<tr>
<td>Xapian::TermGenerator</td>
<td>201</td>
</tr>
<tr>
<td>Xapian::TermIterator</td>
<td>205</td>
</tr>
<tr>
<td>Xapian::Utf8Iterator</td>
<td>216</td>
</tr>
<tr>
<td>Xapian::ValueIterator</td>
<td>224</td>
</tr>
<tr>
<td>Xapian::ValueRangeProcessor</td>
<td>237</td>
</tr>
<tr>
<td>Xapian::StringValueRangeProcessor</td>
<td>199</td>
</tr>
<tr>
<td>Xapian::DateValueRangeProcessor</td>
<td>70</td>
</tr>
<tr>
<td>Xapian::NumberValueRangeProcessor</td>
<td>160</td>
</tr>
<tr>
<td>Xapian::Weight</td>
<td>245</td>
</tr>
</tbody>
</table>
### 3.1 Class Hierarchy

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::BM25Weight</td>
<td>36</td>
</tr>
<tr>
<td>Xapian::BoolWeight</td>
<td>40</td>
</tr>
<tr>
<td>Xapian::TfIdfWeight</td>
<td>207</td>
</tr>
<tr>
<td>Xapian::TradWeight</td>
<td>211</td>
</tr>
</tbody>
</table>
## Chapter 4

### Class Index

#### 4.1 Class List

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

<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::AssertionError</td>
<td>AssertionError is thrown if a logical assertion inside Xapian fails</td>
<td>35</td>
</tr>
<tr>
<td>Xapian::BM25Weight</td>
<td>Xapian::Weight subclass implementing the BM25 probabilistic formula</td>
<td>36</td>
</tr>
<tr>
<td>Xapian::BoolWeight</td>
<td>Class implementing a &quot;boolean&quot; weighting scheme</td>
<td>40</td>
</tr>
<tr>
<td>Xapian::Compactor</td>
<td>Compact a database, or merge and compact several</td>
<td>43</td>
</tr>
<tr>
<td>Xapian::Database</td>
<td>This class is used to access a database, or a group of databases</td>
<td>46</td>
</tr>
<tr>
<td>Xapian::DatabaseCorruptError</td>
<td>DatabaseCorruptError indicates database corruption was detected</td>
<td>58</td>
</tr>
<tr>
<td>Xapian::DatabaseCreateError</td>
<td>DatabaseCreateError indicates a failure to create a database</td>
<td>59</td>
</tr>
<tr>
<td>Xapian::DatabaseError</td>
<td>DatabaseError indicates some sort of database related error</td>
<td>61</td>
</tr>
<tr>
<td>Xapian::DatabaseLockError</td>
<td>DatabaseLockError indicates failure to lock a database</td>
<td>62</td>
</tr>
<tr>
<td>Xapian::DatabaseModifiedError</td>
<td>DatabaseModifiedError indicates a database was modified</td>
<td>64</td>
</tr>
<tr>
<td>Xapian::DatabaseOpeningError</td>
<td>DatabaseOpeningError indicates failure to open a database</td>
<td>66</td>
</tr>
<tr>
<td>Xapian::DatabaseVersionError</td>
<td>DatabaseVersionError indicates that a database is in an unsupported format</td>
<td>68</td>
</tr>
<tr>
<td>Xapian::DateValueRangeProcessor</td>
<td>Handle a date range</td>
<td>70</td>
</tr>
</tbody>
</table>
Xapian::DecreasingValueWeightPostingSource
   Read weights from a value which is known to decrease as docid
   increases .......................................................... 74
Xapian::DocNotFoundError
   Indicates an attempt to access a document not present in the
database ................................................................. 80
Xapian::Document
   A handle representing a document in a Xapian database ....... 81
Xapian::Enquire
   This class provides an interface to the information retrieval system
   for the purpose of searching ........................................ 87
Xapian::Error
   All exceptions thrown by Xapian are subclasses of Xapian::Error 99
Xapian::ErrorHandler
   Decide if a Xapian::Error exception should be ignored .......... 100
Xapian::ESet
   Class representing an ordered set of expand terms (an ESet) .... 101
Xapian::ESetIterator
   Iterate through terms in the ESet .................................. 102
Xapian::ExpandDecider
   Virtual base class for expand decider functor .................... 104
Xapian::ExpandDeciderAnd
   ExpandDecider subclass which rejects terms using two Expand-
   Deciders ............................................................... 105
Xapian::ExpandDeciderFilterPrefix
   ExpandDecider subclass which restrict terms to a particular prefix 107
Xapian::ExpandDeciderFilterTerms
   ExpandDecider subclass which rejects terms in a specified list ... 108
Xapian::FeatureUnavailableError
   Indicates an attempt to use a feature which is unavailable .... 109
Xapian::FieldProcessor
   Base class for field processors .................................... 111
Xapian::FixedWeightPostingSource
   A posting source which returns a fixed weight for all documents .. 112
Xapian::GreatCircleMetric
   Calculate the great-circle distance between two coordinates on a
   sphere ............................................................... 119
Xapian::InternalError
   InternalError indicates a runtime problem of some sort .......... 121
Xapian::InvalidArgumentException
   InvalidArgumentError indicates an invalid parameter value was
   passed to the API ................................................... 123
Xapian::InvalidOperationException
   InvalidOperationError indicates the API was used in an invalid way 124
Xapian::KeyMaker
   Virtual base class for key making functors ....................... 126
Xapian::LatLongCoord
   A latitude-longitude coordinate .................................. 127
Xapian::LatLongCoords
   A sequence of latitude-longitude coordinates .................... 129
<table>
<thead>
<tr>
<th>Class Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::LatLongCoordsIterator</td>
<td>An iterator across the values in a LatLongCoords object</td>
</tr>
<tr>
<td>Xapian::LatLongDistanceKeyMaker</td>
<td>KeyMaker subclass which sorts by distance from a latitude/longitude</td>
</tr>
<tr>
<td>Xapian::LatLongDistancePostingSource</td>
<td>Posting source which returns a weight based on geospatial distance</td>
</tr>
<tr>
<td>Xapian::LatLongMetric</td>
<td>Base class for calculating distances between two lat/long coordinates</td>
</tr>
<tr>
<td>Xapian::LogicError</td>
<td>The base class for exceptions indicating errors in the program logic</td>
</tr>
<tr>
<td>Xapian::MatchDecider</td>
<td>Base class for matcher decision functor</td>
</tr>
<tr>
<td>Xapian::MatchSpy</td>
<td>Abstract base class for match spies</td>
</tr>
<tr>
<td>Xapian::MSet</td>
<td>A match set (MSet)</td>
</tr>
<tr>
<td>Xapian::MSetIterator</td>
<td>An iterator pointing to items in an MSet</td>
</tr>
<tr>
<td>Xapian::MultiValueKeyMaker</td>
<td>KeyMaker subclass which combines several values</td>
</tr>
<tr>
<td>Xapian::NetworkError</td>
<td>Indicates a problem communicating with a remote database</td>
</tr>
<tr>
<td>Xapian::NetworkTimeoutError</td>
<td>Indicates a timeout expired while communicating with a remote database</td>
</tr>
<tr>
<td>Xapian::NumberValueRangeProcessor</td>
<td>Handle a number range</td>
</tr>
<tr>
<td>Xapian::PositionIterator</td>
<td>Class for iterating over term positions</td>
</tr>
<tr>
<td>Xapian::PostingIterator</td>
<td>Class for iterating over a list of terms</td>
</tr>
<tr>
<td>Xapian::PostingSource</td>
<td>Base class which provides an &quot;external&quot; source of postings</td>
</tr>
<tr>
<td>Xapian::Query</td>
<td>Class representing a query</td>
</tr>
<tr>
<td>Xapian::QueryParser</td>
<td>Build a Xapian::Query object from a user query string</td>
</tr>
<tr>
<td>Xapian::QueryParserError</td>
<td>Indicates a query string can't be parsed</td>
</tr>
<tr>
<td>Xapian::RangeError</td>
<td>RangeError indicates an attempt to access outside the bounds of a container</td>
</tr>
<tr>
<td>Xapian::Registry</td>
<td>Registry for user subclasses</td>
</tr>
<tr>
<td>Xapian::RSet</td>
<td>A relevance set (R-Set)</td>
</tr>
<tr>
<td>Xapian::RuntimeError</td>
<td>The base class for exceptions indicating errors only detectable at runtime</td>
</tr>
</tbody>
</table>
Xapian::SerialisationError
   Indicates an error in the std::string serialisation of an object . . . . . . . 191
Xapian::SimpleStopper
   Simple implementation of Stopper class - this will suit most users . . 193
Xapian::Stem
   Class representing a stemming algorithm . . . . . . . . . . . . . . 194
Xapian::StemImplementation
   Class representing a stemming algorithm implementation . . . . . . 197
Xapian::Stopper
   Base class for stop-word decision functor . . . . . . . . . . . . . . 198
Xapian::StringValueRangeProcessor
   Handle a string range . . . . . . . . . . . . . . . . . . . . . . . . 199
Xapian::TermGenerator
   Parses a piece of text and generate terms . . . . . . . . . . . . . . 201
Xapian::TermIterator
   Class for iterating over a list of terms . . . . . . . . . . . . . . . . 205
Xapian::TfIdfWeight
   Xapian::Weight subclass implementing the tf-idf weighting scheme . 207
Xapian::TradWeight
   Xapian::Weight subclass implementing the traditional probabilistic
   formula . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 211
Xapian::UnimplementedError
   UnimplementedError indicates an attempt to use an unimplemented
   feature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 214
Xapian::Utf8Iterator
   An iterator which returns Unicode character values from a UTF-8
   encoded string . . . . . . . . . . . . . . . . . . . . . . . . . . . 216
Xapian::ValueCountMatchSpy
   Class for counting the frequencies of values in the matching docu-
   ments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 220
Xapian::ValueIterator
   Class for iterating over document values . . . . . . . . . . . . . . . . 224
Xapian::ValueMapPostingSource
   A posting source which looks up weights in a map using values as
   the key . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 227
Xapian::ValuePostingSource
   A posting source which generates weights from a value slot . . . . 231
Xapian::ValueRangeProcessor
   Base class for value range processors . . . . . . . . . . . . . . . . . 237
Xapian::ValueSetMatchDecider
   MatchDecider filtering results based on whether document values
   are in a user-defined set . . . . . . . . . . . . . . . . . . . . . . . . 239
Xapian::ValueWeightPostingSource
   A posting source which reads weights from a value slot . . . . . . . 241
Xapian::Weight
   Abstract base class for weighting schemes . . . . . . . . . . . . . . 245
Xapian::WritableDatabase
   This class provides read/write access to a database . . . . . . . 251
Chapter 5

File Index

5.1 File List

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

- xapian/error.h
  - Hierarchy of classes which Xapian can throw as exceptions . . . . . 263
- xapian/version.h
  - Define preprocessor symbols for the library version . . . . . . . . . 264
- xapian.h
  - Public interfaces for the Xapian library . . . . . . . . . . . . . . 265
- xapian/attributes.h
  - Compiler attribute macros . . . . . . . . . . . . . . . . . . . . 266
- xapian/compactor.h
  - Compact a database, or merge and compact several . . . . . . . . . 267
- xapian/database.h
  - API for working with Xapian databases . . . . . . . . . . . . . . 267
- xapian/dbfactory.h
  - Factory functions for constructing Database and WritableDatabase objects . . . . . . . . . . . . . . . . . . . . . . . . . . . . 268
- xapian/document.h
  - API for working with documents . . . . . . . . . . . . . . . . . . 270
- xapian/enquire.h
  - API for running queries . . . . . . . . . . . . . . . . . . . . . . 270
- xapian/errorhandler.h
  - Decide if a Xapian::Error exception should be ignored . . . . . . . 271
- xapian/expanddecider.h
  - Allow rejection of terms during ESet generation . . . . . . . . . . 271
- xapian/geospatial.h
  - Geospatial search support routines . . . . . . . . . . . . . . . . 272
- xapian/keymaker.h
  - Build key strings for MSet ordering or collapsing . . . . . . . . . 273
- xapian/matchspy.h
  - MatchSpy implementation . . . . . . . . . . . . . . . . . . . . . 274
xapian/positioniterator.h
Class for iterating over term positions . . . . . . . . . . . . . . . . 274
xapian/postingiterator.h
Class for iterating over a list of document ids . . . . . . . . . . . . 275
xapian/postingsource.h
External sources of posting information . . . . . . . . . . . . . . . 275
xapian/query.h
Xapian::Query API class . . . . . . . . . . . . . . . . . . . . . . 276
xapian/queryparser.h
Parsing a user query string to build a Xapian::Query object . . . . 276
xapian/registry.h
Class for looking up user subclasses during unserialisation . . . . 277
xapian/stem.h
Stemming algorithms . . . . . . . . . . . . . . . . . . . . . . . . . 278
xapian/termgenerator.h
Parse free text and generate terms . . . . . . . . . . . . . . . . . . 278
xapian/termiterator.h
Class for iterating over a list of terms . . . . . . . . . . . . . . . . 279
xapian/types.h
Typedefs for Xapian . . . . . . . . . . . . . . . . . . . . . . . . . . 280
xapian/unicode.h
Unicode and UTF-8 related classes and functions . . . . . . . . . . 281
xapian/valueiterator.h
Class for iterating over document values . . . . . . . . . . . . . . . 282
xapian/valuesetmatchdecider.h
MatchDecider subclass for filtering results by value . . . . . . . . . 282
xapian/weight.h
Weighting scheme API . . . . . . . . . . . . . . . . . . . . . . . . 283
Chapter 6

Namespace Documentation

6.1 Xapian Namespace Reference

The Xapian namespace contains public interfaces for the Xapian library.

Namespaces

- namespace Auto
  Database factory functions which determine the database type automatically.
- namespace Brass
  Database factory functions for the brass backend.
- namespace Chert
  Database factory functions for the chert backend.
- namespace InMemory
  Database factory functions for the inmemory backend.
- namespace Remote
  Database factory functions for the remote backend.
- namespace Unicode
  Functions associated with handling Unicode characters.

Classes

- class Error
  All exceptions thrown by Xapian are subclasses of Xapian::Error.
- class LogicError
  The base class for exceptions indicating errors in the program logic.
- class RuntimeError
  The base class for exceptions indicating errors only detectable at runtime.
- class AssertionError
AssertionError is thrown if a logical assertion inside Xapian fails.

- class InvalidArgumentError
  InvalidArgumentError indicates an invalid parameter value was passed to the API.

- class InvalidOperationError
  InvalidOperationError indicates the API was used in an invalid way.

- class UnimplementedError
  UnimplementedError indicates an attempt to use an unimplemented feature.

- class DatabaseError
  DatabaseError indicates some sort of database related error.

- class DatabaseCorruptError
  DatabaseCorruptError indicates database corruption was detected.

- class DatabaseCreateError
  DatabaseCreateError indicates a failure to create a database.

- class DatabaseLockError
  DatabaseLockError indicates failure to lock a database.

- class DatabaseModifiedError
  DatabaseModifiedError indicates a database was modified.

- class DatabaseOpeningError
  DatabaseOpeningError indicates failure to open a database.

- class DatabaseVersionError
  DatabaseVersionError indicates that a database is in an unsupported format.

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

- class FeatureUnavailableError
  Indicates an attempt to use a feature which is unavailable.

- class InternalError
  InternalError indicates a runtime problem of some sort.

- class NetworkError
  Indicates a problem communicating with a remote database.

- class NetworkTimeoutError
  Indicates a timeout expired while communicating with a remote database.

- class QueryParserError
  Indicates a query string can't be parsed.

- class SerialisationError
  Indicates an error in the std::string serialisation of an object.

- class RangeError
  RangeError indicates an attempt to access outside the bounds of a container.

- class Compactor
  Compact a database, or merge and compact several.

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

- class WritableDatabase
  This class provides read/write access to a database.
• class Document
  A handle representing a document in a Xapian database.
• class MSet
  A match set (MSet).
• class MSetIterator
  An iterator pointing to items in an MSet.
• class ESet
  Class representing an ordered set of expand terms (an ESet).
• class ESetIterator
  Iterate through terms in the ESet.
• class RSet
  A relevance set (R-Set).
• class MatchDecider
  Base class for matcher decision functor.
• class Enquire
  This class provides an interface to the information retrieval system for the purpose of searching.
• class 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 ExpandDeciderFilterPrefix
  ExpandDecider subclass which restrict terms to a particular prefix.
• struct LatLongCoord
  A latitude-longitude coordinate.
• class LatLongCoordsIterator
  An iterator across the values in a LatLongCoords object.
• class LatLongCoords
  A sequence of latitude-longitude coordinates.
• class LatLongMetric
  Base class for calculating distances between two lat/long coordinates.
• class GreatCircleMetric
  Calculate the great-circle distance between two coordinates on a sphere.
• class LatLongDistancePostingSource
  Posting source which returns a weight based on geospatial distance.
• class LatLongDistanceKeyMaker
  KeyMaker subclass which sorts by distance from a latitude/longitude.
• class KeyMaker
  Virtual base class for key making functors.
• class MultiValueKeyMaker
  KeyMaker subclass which combines several values.
• class MatchSpy
  Abstract base class for match spies.
• class ValueCountMatchSpy
  Class for counting the frequencies of values in the matching documents.
• class PositionIterator
  Class for iterating over term positions.
• class PostingIterator
  Class for iterating over a list of terms.
• class PostingSource
  Base class which provides an "external" source of postings.
• class ValuePostingSource
  A posting source which generates weights from a value slot.
• class ValueWeightPostingSource
  A posting source which reads weights from a value slot.
• class DecreasingValueWeightPostingSource
  Read weights from a value which is known to decrease as docid increases.
• class ValueMapPostingSource
  A posting source which looks up weights in a map using values as the key.
• class FixedWeightPostingSource
  A posting source which returns a fixed weight for all documents.
• 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.
• struct FieldProcessor
  Base class for field processors.
• class QueryParser
  Build a Xapian::Query object from a user query string.
• class Registry
  Registry for user subclasses.
• struct StemImplementation
Class representing a stemming algorithm implementation.

• class Stem
  Class representing a stemming algorithm.

• class TermGenerator
  Parses a piece of text and generate terms.

• class TermIterator
  Class for iterating over a list of terms.

• class Utf8Iterator
  An iterator which returns Unicode character values from a UTF-8 encoded string.

• class ValueIterator
  Class for iterating over document values.

• class ValueSetMatchDecider
  MatchDecider filtering results based on whether document values are in a user-defined set.

• class Weight
  Abstract base class for weighting schemes.

• class BoolWeight
  Class implementing a "boolean" weighting scheme.

• class TfIdfWeight
  Xapian::Weight subclass implementing the tf-idf weighting scheme.

• class BM25Weight
  Xapian::Weight subclass implementing the BM25 probabilistic formula.

• class TradWeight
  Xapian::Weight subclass implementing the traditional probabilistic formula.

Typedefs

• typedef unsigned doccount
  A count of documents.

• typedef int doccount_diff
  A signed difference between two counts of documents.

• typedef unsigned docid
  A unique identifier for a document.

• typedef double doclength
  A normalised document length.

• typedef int percent
  The percentage score for a document in an MSet.

• typedef unsigned termcount
  A counts of terms.

• typedef int termcount_diff
  A signed difference between two counts of terms.

• typedef unsigned termpos
  A term position within a document or query.
• typedef int termpos_diff
  * A signed difference between two term positions.
• typedef unsigned timeout
  * A timeout value in milliseconds.
• typedef unsigned valueno
  * The number for a value slot in a document.
• typedef int valueno_diff
  * A signed difference between two value slot numbers.
• typedef double weight
  * The weight of a document or term.

Functions

• bool operator== (const MSetIterator &a, const MSetIterator &b)
  * Equality test for MSetIterator objects.
• bool operator!= (const MSetIterator &a, const MSetIterator &b)
  * Inequality test for MSetIterator objects.
• bool operator== (const ESetIterator &a, const ESetIterator &b)
  * Equality test for ESetIterator objects.
• bool operator!= (const ESetIterator &a, const ESetIterator &b)
  * Inequality test for ESetIterator objects.
• double miles_to_metres (double miles)
  * Convert from miles to metres.
• double metres_to_miles (double metres)
  * Convert from metres to miles.
• bool operator!= (const LatLongCoordsIterator &a, const LatLongCoordsIterator &b)
  * Inequality test for LatLongCoordsIterator objects.
• bool operator== (const PositionIterator &a, const PositionIterator &b)
  * Equality test for PositionIterator objects.
• bool operator!= (const PositionIterator &a, const PositionIterator &b)
  * Inequality test for PositionIterator objects.
• bool operator== (const PostingIterator &a, const PostingIterator &b)
  * Equality test for PostingIterator objects.
• bool operator!= (const PostingIterator &a, const PostingIterator &b)
  * Inequality test for PostingIterator objects.
• std::string sortable_serialise (double value)
  * Convert a floating point number to a string, preserving sort order.
• double sortable_unserialise (const std::string &value)
  * Convert a string encoded using sortable_serialise back to a floating point number.
6.1 Xapian Namespace Reference

Inequality test for TermIterator objects.

- bool operator== (const ValueIterator &a, const ValueIterator &b)
  Equality test for ValueIterator objects.

- bool operator!= (const ValueIterator &a, const ValueIterator &b)
  Inequality test for ValueIterator objects.

- const char * version_string()
  Report the version string of the library which the program is linked with.

- int major_version()
  Report the major version of the library which the program is linked with.

- int minor_version()
  Report the minor version of the library which the program is linked with.

- int revision()
  Report the revision of the library which the program is linked with.

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 int DBCHECK_SHORT_TREE = 1
  Show a short-format display of the B-tree contents.

- const int DBCHECK_FULL_TREE = 2
  Show a full display of the B-tree contents.

- const int DBCHECK_SHOW_BITMAP = 4
  Show the bitmap for the B-tree.

- const int DBCHECK_SHOW_STATS = 8
  Show statistics for the B-tree.

- const int DBCHECK_FIX = 16
  Fix problems.

- const valuenum BAD_VALUENO = 0xffffffff
  Reserved value to indicate "no valuenum".

6.1.1 Detailed Description

The Xapian namespace contains public interfaces for the Xapian library.
6.1.2 Typedef Documentation

6.1.2.1 typedef unsigned Xapian::doccount

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

6.1.2.2 typedef int Xapian::doccount_diff

A signed difference between two counts of documents.
This is used by the Xapian classes which are STL containers of documents for
"difference_type".

6.1.2.3 typedef unsigned Xapian::docid

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

6.1.2.4 typedef double Xapian::doclength

A normalised document length.
The normalised document length is the document length divided by the average docu-
ment length in the database.

6.1.2.5 typedef int Xapian::percent

The percentage score for a document in an MSet.

Deprecated This type is deprecated as of Xapian 1.3.0 - use the standard type int
instead, which should work with older Xapian too.

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 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".

6.1.2.9 typedef unsigned Xapian::timeout

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

Deprecated This type is deprecated as of Xapian 1.3.0 - use the standard POSIX type useconds_t instead, which should work with older Xapian too.

6.1.2.10 typedef unsigned Xapian::valueno

The number for a value slot in a document.
Value slot numbers are unsigned and (currently) a 32-bit quantity, with Xapian::BAD_VALUENO being represented by the largest possible value. Therefore value slots 0 to 0xFFFFFFFFF are available for use.

6.1.2.11 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".

6.1.2.12 typedef double Xapian::weight

The weight of a document or term.

Deprecated This type is deprecated as of Xapian 1.3.0 - use the standard C++ type double instead, which should work with older Xapian too.

6.1.3 Function Documentation
6.1.3.1 int Xapian::major_version()

Report the major version of the library which the program is linked with.
This may be different to the version compiled against (given by XAPIAN_MAJOR_VERSION) if shared libraries are being used.

6.1.3.2 double Xapian::metres_to_miles(double metres) [inline]

Convert from metres to miles.
Experimental - see http://xapian.org/docs/deprecation#experimental-features

6.1.3.3 double Xapian::miles_to_metres(double miles) [inline]

Convert from miles to metres.
Experimental - see http://xapian.org/docs/deprecation#experimental-features

6.1.3.4 int Xapian::minor_version()

Report the minor version of the library which the program is linked with.
This may be different to the version compiled against (given by XAPIAN_MINOR_VERSION) if shared libraries are being used.

6.1.3.5 int Xapian::revision()

Report the revision of the library which the program is linked with.
This may be different to the version compiled against (given by XAPIAN_REVISION) if shared libraries are being used.

6.1.3.6 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.

Parameters

- **value**: The number to serialise.

### 6.1.3.7 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.

Parameters

- **value**: The serialised string to decode.

### 6.1.3.8 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.

### 6.1.4 Variable Documentation

#### 6.1.4.1 const valueno Xapian::BAD_VALUENO = 0xffffffff

Reserved value to indicate "no valueno".

#### 6.1.4.2 const int Xapian::DB_CREATE = 2

Create a new database; fail if db exists.

#### 6.1.4.3 const int Xapian::DB_CREATE_OR_OPEN = 1

Open for read/write; create if no db exists.
6.1.4.4 const int Xapian::DB_CREATE_OR_OVERWRITE = 3

Overwrite existing db; create if none exists.

6.1.4.5 const int Xapian::DB_OPEN = 4

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

6.1.4.6 const int Xapian::DBCHECK_FIX = 16

Fix problems.
Currently this is supported for chert, and will:
∗ regenerate the "iamchert" file if it isn’t valid (so if it is lost, you can just create it empty
and then “fix problems”).
∗ regenerates base files (currently the algorithm for finding the root block may not work if
there was a change partly written but not committed).
For use with Xapian::Database::check().

6.1.4.7 const int Xapian::DBCHECK_FULL_TREE = 2

Show a full display of the B-tree contents.
For use with Xapian::Database::check().

6.1.4.8 const int Xapian::DBCHECK_SHORT_TREE = 1

Show a short-format display of the B-tree contents.
For use with Xapian::Database::check().

6.1.4.9 const int Xapian::DBCHECK_SHOW_BITMAP = 4

Show the bitmap for the B-tree.
For use with Xapian::Database::check().

6.1.4.10 const int Xapian::DBCHECK_SHOW_STATS = 8

Show statistics for the B-tree.
For use with Xapian::Database::check().
6.2 Xapian::Auto Namespace Reference

Database factory functions which determine the database type automatically.

Functions

- Database open_stub (const std::string &file)
  Construct a Database object for a stub database file.
- WritableDatabase open_stub (const std::string &file, int action)
  Construct a WritableDatabase object for a stub database file.

6.2.1 Detailed Description

Database factory functions which determine the database type automatically.

6.2.2 Function Documentation

6.2.2.1 Database Xapian::Auto::open_stub ( const std::string & file )

Construct a Database object for a stub database file.

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

Parameters

| file | pathname of the stub database file. |

6.2.2.2 WritableDatabase Xapian::Auto::open_stub ( const std::string & file, int action )

Construct a WritableDatabase object for a stub database file.

The stub database file must contain serialised parameters for exactly one database.

Parameters

| file | pathname of the stub database file. |
|      |                           |
| action | determines handling of existing/non-existing database: |
|        | • Xapian::DB_CREATE fail if database already exist, otherwise create new database. |
|        | • Xapian::DB_CREATE_OR_OPEN open existing database, or create new database if none exists. |
|        | • Xapian::DB_CREATE_OR_OVERWRITE overwrite existing database, or create new database if none exists. |
|        | • Xapian::DB_OPEN open existing database, failing if none exists. |
6.3 Xapian::Brass Namespace Reference

Database factory functions for the brass backend.

Functions

- **Database open** (const std::string &dir)
  
  Construct a Database object for read-only access to a Brass database.

- **WritableDatabase open** (const std::string &dir, int action, int block_size=8192)
  
  Construct a Database object for update access to a Brass database.

### 6.3.1 Detailed Description

Database factory functions for the brass backend.

### 6.3.2 Function Documentation

#### 6.3.2.1 Database Xapian::Brass::open ( const std::string & dir )

Construct a Database object for read-only access to a Brass database.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir</td>
<td>pathname of the directory containing the database.</td>
</tr>
</tbody>
</table>

#### 6.3.2.2 WritableDatabase Xapian::Brass::open ( const std::string & dir, int action, int block_size = 8192 )

Construct a Database object for update access to a Brass database.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir</td>
<td>pathname of the directory containing the database.</td>
</tr>
<tr>
<td>action</td>
<td>determines handling of existing/non-existing database:</td>
</tr>
<tr>
<td></td>
<td>- Xapian::DB_CREATE fail if database already exist, otherwise create new database.</td>
</tr>
<tr>
<td></td>
<td>- Xapian::DB_CREATE_OR_OPEN open existing database, or create new database if none exists.</td>
</tr>
<tr>
<td></td>
<td>- Xapian::DB_CREATE_OR_OVERWRITE overwrite existing database, or create new database if none exists.</td>
</tr>
<tr>
<td></td>
<td>- Xapian::DB_OPEN open existing database, failing if none exists.</td>
</tr>
</tbody>
</table>
### 6.4 Xapian::Chert Namespace Reference

**Database** factory functions for the chert backend.

#### Functions

- **Database open** (const std::string &dir)
  
  Construct a **Database** object for read-only access to a Chert database.

- **WritableDatabase open** (const std::string &dir, int action, int block_size=8192)
  
  Construct a **Database** object for update access to a Chert database.

#### 6.4.1 Detailed Description

**Database** factory functions for the chert backend.

#### 6.4.2 Function Documentation

**6.4.2.1 Database Xapian::Chert::open ( const std::string & dir )**

Construct a **Database** object for read-only access to a Chert database.

**Parameters**

- dir pathname of the directory containing the database.

**6.4.2.2 WritableDatabase Xapian::Chert::open ( const std::string & dir, int action, int block_size = 8192 )**

Construct a **Database** object for update access to a Chert database.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir</td>
<td>pathname of the directory containing the database.</td>
</tr>
<tr>
<td>action</td>
<td>determines handling of existing/non-existing database:</td>
</tr>
<tr>
<td>• Xapian::DB_CREATE fail if database already exist, otherwise create new database.</td>
<td></td>
</tr>
<tr>
<td>• Xapian::DB_CREATE_OR_OPEN open existing database, or create new database if none exists.</td>
<td></td>
</tr>
<tr>
<td>• Xapian::DB_CREATE_OR_OVERWRITE overwrite existing database, or create new database if none exists.</td>
<td></td>
</tr>
<tr>
<td>• Xapian::DB_OPEN open existing database, failing if none exists.</td>
<td></td>
</tr>
<tr>
<td>block_size</td>
<td>the Btree blocksize to use (in bytes), which must be a power of two between 2048 and 65536 (inclusive). The default (also used if an invalid value if passed) is 8192 bytes. This parameter is ignored when opening an existing database.</td>
</tr>
</tbody>
</table>

6.5 Xapian::InMemory Namespace Reference

Database factory functions for the inmemory backend.

Functions

- WritableDatabase open ()

  Construct a WritableDatabase object for a new, empty InMemory database.

6.5.1 Detailed Description

Database factory functions for the inmemory backend.

6.5.2 Function Documentation

6.5.2.1 WritableDatabase Xapian::InMemory::open ( )

Construct a WritableDatabase object for a new, empty InMemory database.

Only a writable InMemory database can be created, since a read-only one would always remain empty.

6.6 Xapian::Remote Namespace Reference

Database factory functions for the remote backend.
Functions

- **Database open** (const std::string & host, unsigned int port, useconds_t timeout=10000, useconds_t connect_timeout=10000)

  Construct a **Database** object for read-only access to a remote database accessed via a TCP connection.

- **WritableDatabase open_writable** (const std::string & host, unsigned int port, useconds_t timeout=0, useconds_t connect_timeout=10000)

  Construct a **WritableDatabase** object for update access to a remote database accessed via a TCP connection.

- **Database open** (const std::string & program, const std::string & args, useconds_t timeout=10000)

  Construct a **Database** object for read-only access to a remote database accessed via a program.

- **WritableDatabase open_writable** (const std::string & program, const std::string & args, useconds_t timeout=0)

  Construct a **WritableDatabase** object for update access to a remote database accessed via a program.

### 6.6.1 Detailed Description

**Database** factory functions for the remote backend.

### 6.6.2 Function Documentation

#### 6.6.2.1 Database Xapian::Remote::open

```cpp
(const std::string & host, unsigned int port,
 useconds_t timeout = 10000, useconds_t connect_timeout = 10000 )
```

Construct a **Database** object for read-only access to a remote database accessed via a TCP connection.

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

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>host</code></td>
<td>hostname to connect to.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>port number to connect to.</td>
</tr>
<tr>
<td><code>timeout</code></td>
<td>timeout in milliseconds. If this timeout is exceeded for any individual</td>
</tr>
<tr>
<td></td>
<td>operation on the remote database then <strong>Xapian::NetworkTimeoutError</strong></td>
</tr>
<tr>
<td></td>
<td>is thrown. A timeout of 0 means don’t timeout. (Default is 10000ms, which</td>
</tr>
<tr>
<td></td>
<td>is 10 seconds).</td>
</tr>
<tr>
<td><code>connect_timeout</code></td>
<td>timeout to use when connecting to the server. If this timeout is exceeded</td>
</tr>
<tr>
<td></td>
<td>then <strong>Xapian::NetworkTimeoutError</strong> is thrown. A timeout of 0 means don’t</td>
</tr>
<tr>
<td></td>
<td>timeout. (Default is 10000ms, which is 10 seconds).</td>
</tr>
</tbody>
</table>
6.6.2.2 Database Xapian::Remote::open ( const std::string & program, const std::string & args, useconds_t timeout = 10000 )

Construct a Database object for read-only access to a remote database accessed via a program.

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

Parameters

<table>
<thead>
<tr>
<th>program</th>
<th>the external program to run.</th>
</tr>
</thead>
<tbody>
<tr>
<td>args</td>
<td>space-separated list of arguments to pass to program.</td>
</tr>
<tr>
<td>timeout</td>
<td>timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then Xapian::NetworkTimeoutError is thrown. A timeout of 0 means don't timeout. (Default is 10000ms, which is 10 seconds).</td>
</tr>
</tbody>
</table>

6.6.2.3 WritableDatabase Xapian::Remote::open_writable ( const std::string & host, unsigned int port, useconds_t timeout = 0, useconds_t connect_timeout = 10000 )

Construct a WritableDatabase object for update access to a remote database accessed via a TCP connection.

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

Parameters

<table>
<thead>
<tr>
<th>host</th>
<th>hostname to connect to.</th>
</tr>
</thead>
<tbody>
<tr>
<td>port</td>
<td>port number to connect to.</td>
</tr>
<tr>
<td>timeout</td>
<td>timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then Xapian::NetworkTimeoutError is thrown. (Default is 0, which means don't timeout).</td>
</tr>
<tr>
<td>connect_timeout</td>
<td>timeout to use when connecting to the server. If this timeout is exceeded then Xapian::NetworkTimeoutError is thrown. A timeout of 0 means don’t timeout. (Default is 10000ms, which is 10 seconds).</td>
</tr>
</tbody>
</table>

6.6.2.4 WritableDatabase Xapian::Remote::open_writable ( const std::string & program, const std::string & args, useconds_t timeout = 0 )

Construct a WritableDatabase object for update access to a remote database accessed via a program.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>program</td>
<td>the external program to run.</td>
</tr>
<tr>
<td>args</td>
<td>space-separated list of arguments to pass to program.</td>
</tr>
<tr>
<td>timeout</td>
<td>timeout in milliseconds. If this timeout is exceeded for any individual operation on the remote database then Xapian::NetworkTimeoutError is thrown. (Default is 0, which means don't timeout).</td>
</tr>
</tbody>
</table>

### 6.7 Xapian::Unicode Namespace Reference

Functions associated with handling Unicode characters.

**Enumerations**

- enum **category**

  Each Unicode character is in exactly one of these categories.

**Functions**

- unsigned **nonasciito_utf8** (unsigned ch, char *buf)

  Convert a single non-ASCII Unicode character to UTF-8.

- unsigned **to_utf8** (unsigned ch, char *buf)

  Convert a single Unicode character to UTF-8.

- void **append_utf8** (std::string &s, unsigned ch)

  Append the UTF-8 representation of a single Unicode character to a std::string.

- **category get_category** (unsigned ch)

  Return the category which a given Unicode character falls into.

- bool **is_wordchar** (unsigned ch)

  Test if a given Unicode character is "word character".

- bool **is_whitespace** (unsigned ch)

  Test if a given Unicode character is a whitespace character.

- bool **is_currency** (unsigned ch)

  Test if a given Unicode character is a currency symbol.

- unsigned **tolower** (unsigned ch)

  Convert a Unicode character to lowercase.

- unsigned **toupper** (unsigned ch)

  Convert a Unicode character to uppercase.

- std::string **tolower** (const std::string &term)

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

- std::string **toupper** (const std::string &term)

  Convert a UTF-8 std::string to uppercase.
6.7.1 Detailed Description

Functions associated with handling Unicode characters.

6.7.2 Enumeration Type Documentation

6.7.2.1 enum Xapian::Unicode::category

Each Unicode character is in exactly one of these categories.

6.7.3 Function Documentation

6.7.3.1 unsigned Xapian::Unicode::nonascii_to_utf8 ( unsigned ch, char * buf )

Convert a single non-ASCII Unicode character to UTF-8.

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

Parameters

<table>
<thead>
<tr>
<th>ch</th>
<th>The character (which must be &gt; 128) to write to buf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>buf</td>
<td>The buffer to write the character to - it must have space for (at least) 4 bytes.</td>
</tr>
</tbody>
</table>

Returns

The length of the resultant UTF-8 character in bytes.

Referenced by to_utf8().

6.7.3.2 unsigned Xapian::Unicode::to_utf8 ( unsigned ch, char * buf ) [inline]

Convert a single Unicode character to UTF-8.

Parameters

<table>
<thead>
<tr>
<th>ch</th>
<th>The character to write to buf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>buf</td>
<td>The buffer to write the character to - it must have space for (at least) 4 bytes.</td>
</tr>
</tbody>
</table>

Returns

The length of the resultant UTF-8 character in bytes.

References nonascii_to_utf8().

Referenced by append_utf8().


Chapter 7

Class Documentation

7.1 Xapian::AssertionError Class Reference

AssertionError is thrown if a logical assertion inside Xapian fails.

Inheritance diagram for Xapian::AssertionError:

```
Xapian::AssertionError
Xapian::LogicError
Xapian::Error
```

Public Member Functions

- **AssertionError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.
- **AssertionError** (const std::string &msg_, int errno_)
Detailed Description

AssertionError is thrown if a logical assertion inside Xapian fails.

In a debug build of Xapian, a failed assertion in the core library code will cause -
AssertionError to be thrown.

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

Constructor & Destructor Documentation

7.1.2.1 Xapian::AssertionError::AssertionError ( const std::string & msg_, const std::string & context_ = std::string(), int errno_ = 0 ) [inline, explicit]

General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

7.1.2.2 Xapian::AssertionError::AssertionError ( const std::string & msg_, int errno_ ) [inline]

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.2 Xapian::BM25Weight Class Reference

Xapian::Weight subclass implementing the BM25 probabilistic formula.
7.2 Xapian::BM25Weight Class Reference

Inheritance diagram for Xapian::BM25Weight:

```
Xapian::Weight
  
Xapian::BM25Weight
```

Public Member Functions

- **BM25Weight** (double k1, double k2, double k3, double b, double min_normlen)
  
  *Constructor a BM25Weight.*

- **std::string name () const**
  
  *Return the name of this weighting scheme.*

- **std::string serialise () const**
  
  *Return this object’s parameters serialised as a single string.*

- **BM25Weight ∗ unserialise (const std::string &s) const**
  
  *Unserialise parameters.*

- **double get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const**
  
  *Calculate the weight contribution for this object’s term to a document.*

- **double get_maxpart () const**
  
  *Return an upper bound on what get_sumpart() can return for any document.*

- **double get_sumextra (Xapian::termcount doclen) const**
  
  *Calculate the term-independent weight component for a document.*

- **double get_maxextra () const**
  
  *Return an upper bound on what get_sumextra() can return for any document.*

7.2.1 Detailed Description

Xapian::Weight subclass implementing the BM25 probabilistic formula.

7.2.2 Constructor & Destructor Documentation
7.2.2.1  Xapian::BM25Weight::BM25Weight ( double k1, double k2, double k3, double b,
     double min_normlen ) [inline]

Construct a BM25Weight.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>A non-negative parameter controlling how influential within-document-frequency (wdf) is. k1=0 means that wdf doesn't affect the weights. The larger k1 is, the more wdf influences the weights. (default 1)</td>
</tr>
<tr>
<td>k2</td>
<td>A non-negative parameter which controls the strength of a correction factor which depends upon query length and normalised document length. k2=0 disable this factor; larger k2 makes it stronger. (default 0)</td>
</tr>
<tr>
<td>k3</td>
<td>A non-negative parameter controlling how influential within-query-frequency (wqf) is. k3=0 means that wqf doesn't affect the weights. The larger k3 is, the more wqf influences the weights. (default 1)</td>
</tr>
<tr>
<td>b</td>
<td>A parameter between 0 and 1, controlling how strong the document length normalisation of wdf is. 0 means no normalisation; 1 means full normalisation. (default 0.5)</td>
</tr>
<tr>
<td>min_normlen</td>
<td>A parameter specifying a minimum value for normalised document length. Normalised document length values less than this will be clamped to this value, helping to prevent very short documents getting large weights. (default 0.5)</td>
</tr>
</tbody>
</table>

7.2.3  Member Function Documentation

7.2.3.1  double Xapian::BM25Weight::get_maxextra () const  [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.2.3.2  double Xapian::BM25Weight::get_maxpart () const  [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.2.3.3  double Xapian::BM25Weight::get_sumextra ( Xapian::termcount doclen )
     const  [virtual]

Calculate the term-independent weight component for a document.
The parameter gives information about the document which may be used in the calculations:

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>doclen</code></td>
<td>The document’s length (unnormalised).</td>
</tr>
</tbody>
</table>

Implements `Xapian::Weight`.

7.2.3.4 `double` `Xapian::BM25Weight::get_sumpart` `(` `Xapian::termcount` `wdf,`  
`Xapian::termcount` `doclen`)` `const` `[virtual]`

Calculate the weight contribution for this object’s term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wdf</code></td>
<td>The within document frequency of the term in the document.</td>
</tr>
<tr>
<td><code>doclen</code></td>
<td>The document’s length (unnormalised).</td>
</tr>
</tbody>
</table>

Implements `Xapian::Weight`.

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

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (`Xapian::BM25Weight` returns "Xapian::BM25Weight" here).

If you don’t want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented from `Xapian::Weight`.

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

Return this object’s parameters serialised as a single string.

If you don’t want to support the remote backend, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Reimplemented from `Xapian::Weight`. 
7.2.3.7 BM25Weight+ Xapian::BM25Weight::unserialise ( const std::string & s ) const
    [virtual]

Unserialise parameters.
This method unserialises parameters serialised by the serialise() method and allocates and returns a new object initialised with them.
If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.
Note that the returned object will be deallocated by Xapian after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.-org/ticket/554#comment:1

Parameters

| s | A string containing the serialised parameters. |

Reimplemented from Xapian::Weight.
The documentation for this class was generated from the following file:
- xapian/weight.h

7.3 Xapian::BoolWeight Class Reference

Class implementing a "boolean" weighting scheme.
Inheritance diagram for Xapian::BoolWeight:
7.3 Xapian::BoolWeight Class Reference

Public Member Functions

- **BoolWeight ()**
  Construct a BoolWeight.
- **std::string name () const**
  Return the name of this weighting scheme.
- **std::string serialise () const**
  Return this object's parameters serialised as a single string.
- **BoolWeight * unserialise (const std::string &s) const**
  Unserialise parameters.
- **double get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const**
  Calculate the weight contribution for this object's term to a document.
- **double get_maxpart () const**
  Return an upper bound on what get_sumpart() can return for any document.
- **double get_sumextra (Xapian::termcount doclen) const**
  Calculate the term-independent weight component for a document.
- **double get_maxextra () const**
  Return an upper bound on what get_sumextra() can return for any document.

7.3.1 Detailed Description

Class implementing a "boolean" weighting scheme.
This weighting scheme gives all documents zero weight.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Xapian::BoolWeight::BoolWeight ( ) [inline]

Construct a BoolWeight.

7.3.3 Member Function Documentation

7.3.3.1 double Xapian::BoolWeight::get_maxextra ( ) const [virtual]

Return an upper bound on what get_sumextra() can return for any document.
This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.
Implements Xapian::Weight.
7.3.3.2 double Xapian::BoolWeight::get_maxpart() const [virtual]

Return an upper bound on what get_sumpart() can return for any document.
This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.
Implements Xapian::Weight.

7.3.3.3 double Xapian::BoolWeight::get_sumextra( Xapian::termcount doclen ) const [virtual]

Calculate the term-independent weight component for a document.
The parameter gives information about the document which may be used in the calculations:

Parameters

| doclen | The document's length (unnormalised). |

Implements Xapian::Weight.

7.3.3.4 double Xapian::BoolWeight::get_sumpart( Xapian::termcount wdf,
Xapian::termcount doclen ) const [virtual]

Calculate the weight contribution for this object's term to a document.
The parameters give information about the document which may be used in the calculations:

Parameters

<table>
<thead>
<tr>
<th>wdf</th>
<th>The within document frequency of the term in the document.</th>
</tr>
</thead>
<tbody>
<tr>
<td>doclen</td>
<td>The document's length (unnormalised).</td>
</tr>
</tbody>
</table>

Implements Xapian::Weight.

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

Return the name of this weighting scheme.
This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.
Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).
If you don’t want to support the remote backend, you can use the default implementation which simply returns an empty string.
7.4 Xapian::Compactor Class Reference

Reimplemented from Xapian::Weight.

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

Return this object's parameters serialised as a single string.
If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.
Reimplemented from Xapian::Weight.

7.3.3.7 BoolWeight* Xapian::BoolWeight::unserialise ( const std::string & s ) const [virtual]

Unserialise parameters.
This method unserialises parameters serialised by the serialise() method and allocates and returns a new object initialised with them.
If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.
Note that the returned object will be deallocated by Xapian after use with "delete". -
If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.-org/ticket/554#comment:1

Parameters

| s | A string containing the serialised parameters. |

Reimplemented from Xapian::Weight.
The documentation for this class was generated from the following file:

- xapian/weight.h

7.4 Xapian::Compactor Class Reference

Compact a database, or merge and compact several.

Public Member Functions

- void set_block_size (size_t block_size)
  
  Set the block size to use for tables in the output database.
- void set_renumber (bool renumber)
  
  Set whether to preserve existing document id values.
44 Class Documentation

- **void set_multipass** (bool multipass)
  
  *Set whether to merge postlists in multiple passes.*

- **void set_compaction_level** (compaction_level compaction)
  
  *Set the compaction level.*

- **void set_destdir** (const std::string &destdir)
  
  *Set where to write the output.*

- **void add_source** (const std::string &srcdir)
  
  *Add a source database.*

- **void compact** ()
  
  *Perform the actual compaction/merging operation.*

- **virtual void set_status** (const std::string &table, const std::string &status)
  
  *Update progress.*

- **virtual std::string resolve_duplicate_metadata** (const std::string &key, size_t num_tags, const std::string tags[])
  
  *Resolve multiple user metadata entries with the same key.*

7.4.1 Detailed Description

Compact a database, or merge and compact several.

7.4.2 Member Function Documentation

7.4.2.1 **void Xapian::Compactor::add_source** ( const std::string & srcdir )

Add a source database.

**Parameters**

| srcdir | The path to the source database to add. |

7.4.2.2 **virtual std::string Xapian::Compactor::resolve_duplicate_metadata** ( const std::string & key, size_t num_tags, const std::string tags[] ) [virtual]

Resolve multiple user metadata entries with the same key.

When merging, if the same user metadata key is set in more than one input, then this method is called to allow this to be resolving in an appropriate way.

The default implementation just returns tags[0].

For multipass this will currently get called multiple times for the same key if there are duplicates to resolve in each pass, but this may change in the future.

**Parameters**

| key | The metadata key with duplicate entries. |
| num_tags | How many tags there are. |
7.4 Xapian::Compactor Class Reference

### 7.4.2.3 void Xapian::Compactor::set_block_size ( size_t block_size )

Set the block size to use for tables in the output database.

**Parameters**

| block_size | The block size to use. Valid block sizes are currently powers of two between 2048 and 65536, with the default being 8192, but the valid sizes and default may change in the future. |

### 7.4.2.4 void Xapian::Compactor::set_compaction_level ( compaction_level compaction )

Set the compaction level.

**Parameters**

| compaction | Available values are: - Xapian::Compactor::STANDARD - Don’t split items unnecessarily. - Xapian::Compactor::FULL - Split items whenever it saves space (the default). - Xapian::Compactor::FULLER - Allow oversize items to save more space (not recommended if you ever plan to update the compacted database). |

### 7.4.2.5 void Xapian::Compactor::set_destdir ( const std::string & destdir )

Set where to write the output.

**Parameters**

| destdir | Output path. This can be the same as an input if that input is a stub database (in which case the database(s) listed in the stub will be compacted to a new database and then the stub will be atomically updated to point to this new database). |

### 7.4.2.6 void Xapian::Compactor::set_multipass ( bool multipass )

Set whether to merge postlists in multiple passes.

**Parameters**

| multipass | If true and merging more than 3 databases, merge the postlists in multiple passes, which is generally faster but requires more disk space for temporary files. By default we don’t do this. |
7.4.2.7 void Xapian::Compactor::set_renumber ( bool renumber )

Set whether to preserve existing document id values.

Parameters

| renumber | The default is true, which means that document ids will be renumbered - currently by applying the same offset to all the document ids in a particular source database. |

If false, then the document ids must be unique over all source databases. Currently the ranges of document ids in each source must not overlap either, though this restriction may be removed in the future.

7.4.2.8 virtual void Xapian::Compactor::set_status ( const std::string & table, const std::string & status ) [virtual]

Update progress.

Subclass this method if you want to get progress updates during compaction. This is called for each table first with empty status, And then one or more times with non-empty status.

The default implementation does nothing.

Parameters

| table | The table currently being compacted. |
| status | A status message. |

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

- xapian/compactor.h

7.5 Xapian::Database Class Reference

This class is used to access a database, or a group of databases.
Inheritance diagram for Xapian::Database:

```
+-----------------+              +------------------+
|                 |              |                  |
|     Xapian::Database     |     Xapian::WritableDatabase     |
|                 |              |                  |
```

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.

- bool reopen ()
  
  Re-open the database.

- virtual void close ()
  
  Close the database.

- virtual std::string get_description () const
  
  Return a string describing this object.

- PostingIterator postlist_begin (const std::string &name) const
  
  An iterator pointing to the start of the postlist for a given term.

- PostingIterator postlist_end (const std::string &) const
  
  Corresponding end iterator to postlist_begin().

- TermIterator termlist_begin (Xapian::docid did) const
  
  An iterator pointing to the start of the termlist for a given document.

- TermIterator termlist_end (Xapian::docid) const
Corresponding end iterator to `termlist_begin()`.

- **bool has_positions () const**
  
  *Does this database have any positional information?*

- **PositionIterator positionlist_begin (Xapian::docid did, const std::string &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::doccount get_value_freq (Xapian::valueno slot) const**
  
  *Return the frequency of a given value slot.*

- **std::string get_value_lower_bound (Xapian::valueno slot) const**
  
  *Get a lower bound on the values stored in the given value slot.*

- **std::string get_value_upper_bound (Xapian::valueno slot) const**
  
  *Get an upper bound on the values stored in the given value slot.*

- **Xapian::termcount get_doclength_lower_bound () const**
  
  *Get a lower bound on the length of a document in this DB.*

- **Xapian::termcount get_doclength_upper_bound () const**
  
  *Get an upper bound on the length of a document in this DB.*

- **Xapian::termcount get_wdf_upper_bound (const std::string &term) const**
  
  *Get an upper bound on the wdf of term term.*

- **Valuelterator valuestream_begin (Xapian::valueno slot) const**
  
  *Return an iterator over the value in slot slot for each document.*

- **Valuelterator valuestream_end (Xapian::valueno) const**
Return end iterator corresponding to `valuestream_begin()`.

- **Xapian::termcount 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 &term) const**
  
  Corresponding end iterator to `synonyms_begin(term)`.

- **Xapian::TermIterator synonym_keys_begin (const std::string &prefix=std::string()) const**
  
  An iterator which returns all terms which have synonyms.

- **Xapian::TermIterator synonym_keys_end (const std::string &prefix=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=std::string()) const**
  
  An iterator which returns all user-specified metadata keys.

- **Xapian::TermIterator metadata_keys_end (const std::string &prefix=std::string()) const**
  
  Corresponding end iterator to `metadata_keys_begin()`.

- **std::string get_uuid () const**
  
  Get a UUID for the database.

### Static Public Member Functions

- **static size_t check (const std::string &path, int opts, std::ostream &out)**
  
  Check the integrity of a database or database table.

- **static size_t check (const std::string &path, int opts)**
  
  Check the integrity of a database or database table.
7.5.1 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

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidArgumentError</td>
<td>will be thrown if an invalid argument is supplied, for example, an unknown database type.</td>
</tr>
<tr>
<td>DatabaseOpeningError</td>
<td>may be thrown if the database cannot be opened (for example, a required file cannot be found).</td>
</tr>
<tr>
<td>DatabaseVersionError</td>
<td>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).</td>
</tr>
</tbody>
</table>

7.5.2 Constructor & Destructor Documentation

7.5.2.1 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. |

7.5.2.2 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.

7.5.2.3 Xapian::Database::Database ( const Database & other )

Copying is allowed.
The internals are reference counted, so copying is cheap.

Parameters

| other | The object to copy. |

7.5.3 Member Function Documentation

7.5.3.1 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. |

### 7.5.3.2 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 remote databases.

Parameters

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

### 7.5.3.3 static size_t

**Xapian::Database::check ( const std::string & path, int opts, std::ostream & out ) [static]**

Check the integrity of a database or database table.

This method is currently experimental, and may change incompatibly or possibly even be removed. Feedback on how well it works and how it might be improved are welcome.

Parameters

| path | Path to database or table |
| opts | Options to use for check |
| out | std::ostream to write output to |

### 7.5.3.4 static size_t

**Xapian::Database::check ( const std::string & path, int opts ) [static]**

Check the integrity of a database or database table.

This method is currently experimental, and may change incompatibly or possibly even be removed. Feedback on how well it works and how it might be improved are welcome.

Parameters

| path | Path to database or table |
| opts | Options to use for check |
7.5.3.5  virtual void Xapian::Database::close() [virtual]

Close the database.

This closes the database and closes all its file handles.

For a WritableDatabase, if a transaction is active it will be aborted, while if no transaction is active commit() will be implicitly called. Also the write lock is released.

Closing a database cannot be undone - in particular, calling reopen() after close() will not reopen it, but will instead throw a Xapian::DatabaseError exception.

Calling close() again on a database which has already been closed has no effect (and doesn’t raise an exception).

After close() has been called, calls to other methods of the database, and to methods of other objects associated with the database, will either:

- behave exactly as they would have done if the database had not been closed (this can only happen if all the required data is cached)
- raise a Xapian::DatabaseError exception indicating that the database is closed.

The reason for this behaviour is that otherwise we’d have to check that the database is still open on every method call on every object associated with a Database, when in many cases they are working on data which has already been loaded and so they are able to just behave correctly.

This method was added in Xapian 1.1.0.

7.5.3.6  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. |

7.5.3.7  Xapian::termcount Xapian::Database::get_doclength_lower_bound() const

Get a lower bound on the length of a document in this DB.

This bound does not include any zero-length documents.
7.5 Xapian::Database Class Reference

7.5.3.8 Xapian::Document Xapian::Database::get_document ( Xapian::docid did ) const

Get a document from the database, given its document id.
This method returns a Xapian::Document object which provides the information about a document.

Parameters

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

Returns

A Xapian::Document object containing the document data

Exceptions

<table>
<thead>
<tr>
<th>Xapian::DocNotFoundError</th>
<th>The document specified could not be found in the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::InvalidArgumentError</td>
<td>did was 0, which is not a valid document id.</td>
</tr>
</tbody>
</table>

7.5.3.9 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. |

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.5.3.10 std::string Xapian::Database::get_spelling_suggestion ( const std::string & word, unsigned max_edit_distance = 2 ) const

Suggest a spelling correction.

Parameters

<table>
<thead>
<tr>
<th>word</th>
<th>The potentially misspelled word.</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_edit_distance</td>
<td>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).</td>
</tr>
</tbody>
</table>

7.5.3.11 std::string Xapian::Database::get_uuid ( ) const

Get a UUID for the database.

The UUID will persist for the lifetime of the database.

Replicas (eg, made with the replication protocol, or by copying all the database files) will have the same UUID. However, copies (made with copydatabase, or xapian-compact) will have different UUIDs.

If the backend does not support UUIDs or this database has no subdatabases, the UUID will be empty.

If this database has multiple sub-databases, the UUID string will contain the UUIDs of all the sub-databases.

7.5.3.12 Xapian::doccount Xapian::Database::get_value_freq ( Xapian::valueno slot ) const

Return the frequency of a given value slot.

This is the number of documents which have a (non-empty) value stored in the slot.

Parameters

| slot | The value slot to examine. |

Exceptions

UnimplementedError  The frequency of the value isn’t available for this database type.

7.5.3.13 std::string Xapian::Database::get_value_lower_bound ( Xapian::valueno slot ) const

Get a lower bound on the values stored in the given value slot.

If there are no values stored in the given value slot, this will return an empty string.
If the lower bound isn’t available for the given database type, this will return the lowest possible bound - the empty string.

Parameters

| slot | The value slot to examine. |

7.5.3.14 std::string Xapian::Database::get_value_upper_bound ( Xapian::valueno slot ) const

Get an upper bound on the values stored in the given value slot.
If there are no values stored in the given value slot, this will return an empty string.

Parameters

| slot | The value slot to examine. |

Exceptions

UnimplementedError The upper bound of the values isn’t available for this database type.

7.5.3.15 void Xapian::Database::keep_alive ( )

Send a "keep-alive" to remote databases to stop them timing out.
Has no effect on non-remote databases.

7.5.3.16 Xapian::TermIterator Xapian::Database::metadata_keys_begin ( const std::string & prefix = std::string() ) const

An iterator which returns all user-specified metadata keys.
When invoked on a Xapian::Database object representing multiple databases, currently only the metadata for the first is considered but this behaviour may change in the future.
If the backend doesn’t support metadata, then this method returns an iterator which compares equal to that returned by metadata_keys_end().

Parameters

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

Exceptions

UnimplementedError will be thrown if the backend implements user-specified metadata, but doesn’t implement iterating its keys (currently this happens for the InMemory backend).
7.5.3.17 void Xapian::Database::operator= ( const Database & other )

Assignment is allowed.
The internals are reference counted, so assignment is cheap.

Parameters

| other | The object to copy. |

7.5.3.18 PostingIterator Xapian::Database::postlist_begin ( const std::string & tname ) const

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

Parameters

| tname | The termname to iterate postings for. 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. |

7.5.3.19 bool 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.

Calling reopen() on a database which has been closed (with close()) will always raise a Xapian::DatabaseError.

Returns

true if the database might have been reopened (if false is returned, the database definitely hasn’t been reopened, which applications may find useful when caching results, etc). In Xapian < 1.3.0, this method did not return a value.

7.5.3.20 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.
7.5.3.21 Xapian::TermIterator Xapian::Database::synonym_keys_begin ( const std::string & prefix = std::string() ) const

An iterator which returns all terms which have synonyms.

Parameters

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

7.5.3.22 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. |

7.5.3.23 bool Xapian::Database::term_exists ( const std::string & tname ) const

Check if a given term exists in the database.

Parameters

| tname | The term to test the existence of. |

Returns

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.

7.5.3.24 TermIterator Xapian::Database::termlist_begin ( Xapian::docid did ) const

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

Parameters

| did | The document id of the document to iterate terms for. |

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

- xapian/database.h
7.6 Xapian::DatabaseCorruptError Class Reference

DatabaseCorruptError indicates database corruption was detected.

Inheritance diagram for Xapian::DatabaseCorruptError:

```
Xapian::Error
   ^
   |   Xapian::RuntimeError
   ^
   |   Xapian::DatabaseError
   ^
   |   Xapian::DatabaseCorruptError
```

Public Member Functions

- **DatabaseCorruptError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  *General purpose constructor.*

- **DatabaseCorruptError** (const std::string &msg_, int errno_)
  
  *Construct from message and errno value.*

7.6.1 Detailed Description

DatabaseCorruptError indicates database corruption was detected.

7.6.2 Constructor & Destructor Documentation
7.7 Xapian::DatabaseCreateError Class Reference

7.6.2.1 Xapian::DatabaseCorruptError::DatabaseCorruptError ( const std::string & msg_, const std::string & context_, = std::string(), int errno_ = 0 ) [inline, explicit]

General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

7.6.2.2 Xapian::DatabaseCorruptError::DatabaseCorruptError ( const std::string & msg_, int errno_ ) [inline]

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

---

7.7 Xapian::DatabaseCreateError Class Reference

DatabaseCreateError indicates a failure to create a database.

---

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
Inheritance diagram for Xapian::DatabaseCreateError:

```
Xapian::Error
  ^
Xapian::RuntimeError
  ^
Xapian::DatabaseError
  ^
Xapian::DatabaseCreateError
```

Public Member Functions

- **DatabaseCreateError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.

- **DatabaseCreateError** (const std::string &msg_, int errno_)
  
  Construct from message and errno value.

7.7.1 Detailed Description

**DatabaseCreateError** indicates a failure to create a database.

7.7.2 Constructor & Destructor Documentation

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

General purpose constructor.
7.8 Xapian::DatabaseError Class Reference

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.8 Xapian::DatabaseError Class Reference

DatabaseError indicates some sort of database related error.

Inheritance diagram for Xapian::DatabaseError:

```
Xapian::Error
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Public Member Functions

- **DatabaseError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  *General purpose constructor.*

- **DatabaseError** (const std::string &msg_, int errno_)
  
  *Construct from message and errno value.*
7.8.1 Detailed Description

`DatabaseError` indicates some sort of database related error.

7.8.2 Constructor & Destructor Documentation

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

General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg_</code></td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td><code>context_</code></td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td><code>errno_</code></td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg_</code></td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td><code>errno_</code></td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.9 Xapian::DatabaseLockError Class Reference

`DatabaseLockError` indicates failure to lock a database.
Inheritance diagram for Xapian::DatabaseLockError:

```
Xapian::Error
  ↓
Xapian::RuntimeError
  ↓
Xapian::DatabaseError
  ↓
Xapian::DatabaseLockError
```

### Public Member Functions

- **DatabaseLockError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  *General purpose constructor.*

- **DatabaseLockError** (const std::string &msg_, int errno_)

  *Construct from message and errno value.*

### 7.9.1 Detailed Description

*DatabaseLockError* indicates failure to lock a database.

### 7.9.2 Constructor & Destructor Documentation

**7.9.2.1 Xapian::DatabaseLockError::DatabaseLockError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0) [inline, explicit]

*General purpose constructor.*
Parameters

<table>
<thead>
<tr>
<th>msg_</th>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

7.9.2 DatabaseLockError::DatabaseLockError (const std::string & msg_, int errno_)

[inline]

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>msg_</th>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.10 Xapian::DatabaseModifiedError Class Reference

DatabaseModifiedError indicates a database was modified.
Inheritance diagram for Xapian::DatabaseModifiedError:

Public Member Functions

- **DatabaseModifiedError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.

- **DatabaseModifiedError** (const std::string &msg_, int errno_)
  
  Construct from message and errno value.

7.10.1 Detailed Description

**DatabaseModifiedError** indicates a database was modified.

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

7.10.2 Constructor & Destructor Documentation
General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>context</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>errno</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h
Inheritance diagram for Xapian::DatabaseOpeningError:

```
Xapian::Error
  ├── Xapian::RuntimeError
  │    └── Xapian::DatabaseError
  │         └── Xapian::DatabaseOpeningError
  │                  └── Xapian::DatabaseVersionError
```

Public Member Functions

- `DatabaseOpeningError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)`
  
  General purpose constructor.

- `DatabaseOpeningError (const std::string &msg_, int errno_)

  Construct from message and errno value.

7.11.1 Detailed Description

`DatabaseOpeningError` indicates failure to open a database.

7.11.2 Constructor & Destructor Documentation

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.11.2.1 Xapian::DatabaseOpeningError::DatabaseOpeningError ( const std::string & msg_,
const std::string & context_ = std::string(), int errno_ = 0 ) [inline, explicit]

General purpose constructor.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

Construct from message and errno value.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.12 Xapian::DatabaseVersionError Class Reference

DatabaseVersionError indicates that a database is in an unsupported format.
Inheritance diagram for Xapian::DatabaseVersionError:

```
    Xapian::Error
       |          |
       |          | Xapian::RuntimeError
       |          |
       |          | Xapian::DatabaseError
       |          |
    Xapian::DatabaseOpeningError
```

Public Member Functions

- `DatabaseVersionError(const std::string &msg_, const std::string &context_=std::string(), int errno_=0)`
  General purpose constructor.
- `DatabaseVersionError(const std::string &msg_, int errno_)`
  Construct from message and errno value.

### 7.12.1 Detailed Description

`DatabaseVersionError` indicates that a database is in an unsupported format.

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

### 7.12.2 Constructor & Destructor Documentation

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

General purpose constructor.

**Parameters**

| msg_       | Message giving details of the error, intended for human consumption. |
| context_   | Optional context information for this error.                     |
| errno_     | Optional errno value associated with this error.                 |

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

Construct from message and errno value.

**Parameters**

| msg_       | Message giving details of the error, intended for human consumption. |
| errno_     | Optional errno value associated with this error.                   |

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

- xapian/error.h

### 7.13 Xapian::DateValueRangeProcessor Class Reference

Handle a date range.
Inheritance diagram for Xapian::DateValueRangeProcessor:

```
Xapian::ValueRangeProcessor
  ↓
Xapian::StringValueRangeProcessor
  ↓
Xapian::DateValueRangeProcessor
```

Public Member Functions

- `DateValueRangeProcessor (Xapian::valueno slot_, bool prefer_mdy_=false, int epoch_year_=1970)`
  Constructor.
- `DateValueRangeProcessor (Xapian::valueno slot_, const std::string &str_, bool prefix_=true, bool prefer_mdy_=false, int epoch_year_=1970)`
  Constructor.
- `DateValueRangeProcessor (Xapian::valueno slot_, const char *str_, bool prefix_=true, bool prefer_mdy_=false, int epoch_year_=1970)`
  Constructor.
- `Xapian::valueno operator() (std::string &begin, std::string &end)`
  Check for a valid date range.

7.13.1 Detailed Description

Handle a date range.
Begin and end must be dates in a recognised format.

7.13.2 Constructor & Destructor Documentation
7.13.2.1 Xapian::DateValueRangeProcessor::DateValueRangeProcessor {
    Xapian::valueno slot_, bool prefer_mdy_ = false, int epoch_year_ = 1970 }
    [inline]

    Constructor.

    Parameters

    | slot | 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). |

7.13.2.2 Xapian::DateValueRangeProcessor::DateValueRangeProcessor {
    Xapian::valueno slot_, const std::string & str_, bool prefix_ = true, bool prefer_mdy_ = false, int epoch_year_ = 1970 }
    [inline]

    Constructor.

    Parameters

    | slot | The value number to return from operator(). |
    | str_ | A string to look for to recognise values as belonging to this date 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). |
    | 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). |

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 dates.

For example, if str_ is "created:" and prefix_ is true, and the range processor has been added to the queryparser, the queryparser will accept "created:1/1/2000..31/12/2001".

7.13.2.3 Xapian::DateValueRangeProcessor::DateValueRangeProcessor {
    Xapian::valueno slot_, const char * str_, bool prefix_ = true, bool prefer_mdy_ = false, int epoch_year_ = 1970 }
    [inline]

    Constructor.
This is like the previous version, but with const char * instead of std::string - we need this overload as otherwise DateValueRangeProcessor(1, "date:")) quietly interprets the second argument as a boolean in preference to std::string. If you want to be compatible with 1.2.12 and earlier, then explicitly convert to std::string, i.e.: DateValueRangeProcessor(1, std::string("date:"))

Parameters

<table>
<thead>
<tr>
<th>slot_</th>
<th>The value number to return from operator().</th>
</tr>
</thead>
<tbody>
<tr>
<td>str_</td>
<td>A string to look for to recognise values as belonging to this date range.</td>
</tr>
<tr>
<td>prefix_</td>
<td>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).</td>
</tr>
<tr>
<td>prefer_mdy_-</td>
<td>Should ambiguous dates be interpreted as month/day/year rather than day/month/year? (default: false)</td>
</tr>
<tr>
<td>epoch_year-</td>
<td>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).</td>
</tr>
</tbody>
</table>

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 dates.

For example, if str_ is "created:" and prefix_ is true, and the range processor has been added to the query parser, the query parser will accept "created:1/1/2000..31/12/2001".

7.13.3 Member Function Documentation

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

Check for a valid date range.

Parameters

<table>
<thead>
<tr>
<th>in, out</th>
<th>begin</th>
<th>The start of the range as specified in the query string by the user. This parameter is a non-const reference so the ValueRangeProcessor can modify it to return the value to start the range with.</th>
</tr>
</thead>
<tbody>
<tr>
<td>in, out</td>
<td>end</td>
<td>The end of the range. This is also a non-const reference so it can be modified.</td>
</tr>
</tbody>
</table>
Returns

If BEGIN..END is a sensible date range, this method modifies them into the format YYYYMMDD and returns the value of slot_ passed at construction time. Otherwise it returns Xapian::BAD_VALUENO.

Reimplemented from Xapian::StringValueRangeProcessor.

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

- xapian/queryparser.h

### 7.14 Xapian::DecreasingValueWeightPostingSource Class Reference

Read weights from a value which is known to decrease as docid increases.

Inheritance diagram for Xapian::DecreasingValueWeightPostingSource:

```
Xapian::PostingSource
  
Xapian::ValuePostingSource
  
Xapian::ValueWeightPostingSource
  
Xapian::DecreasingValueWeightPostingSource
```

#### Public Member Functions

- `double get_weight () const`
  
  Return the weight contribution for the current document.

- `DecreasingValueWeightPostingSource * clone () const`
Clone the posting source.

- std::string name () const
  Name of the posting source class.
- std::string serialise () const
  Serialise object parameters into a string.
- DecreasingValueWeightPostingSource ∗ unserialise (const std::string &s) const
  Create object given string serialisation returned by serialise().
- void init (const Xapian::Database &db_)
  Set this PostingSource to the start of the list of postings.
- void next (double min_wt)
  Advance the current position to the next matching document.
- void skip_to (Xapian::docid min_docid, double min_wt)
  Advance to the specified docid.
- bool check (Xapian::docid min_docid, double min_wt)
  Check if the specified docid occurs.
- std::string get_description () const
  Return a string describing this object.

Protected Member Functions

- void skip_if_in_range (double min_wt)
  Skip the iterator forward if in the decreasing range, and weight is low.

Protected Attributes

- bool items_at_end
  Flag, set to true if there are docs after the end of the range.

7.14.1 Detailed Description

Read weights from a value which is known to decrease as docid increases.

This posting source can be used, like ValueWeightPostingSource, to add a weight contribution to a query based on the values stored in a slot. The values in the slot must be serialised as by sortable_serialise().

However, this posting source is additionally given a range of document IDs, within which the weight is known to be decreasing. i.e., for all documents with ids A and B within this range (including the endpoints), where A is less than B, the weight of A is less than or equal to the weight of B. This can allow the posting source to skip to the end of the range quickly if insufficient weight is left in the posting source for a particular source.

By default, the range is assumed to cover all document IDs.

The ordering property can be arranged at index time, or by sorting an indexed database to produce a new, sorted, database.
7.14.2 Member Function Documentation

7.14.2.1 bool Xapian::DecreasingValueWeightPostingSource::check (  
  Xapian::docid did, double min_wt ) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id did actually exists in the  
database. If it does, it must move to that document id, and return true. If it does not, it  
may either:

• return true, having moved to a definite position (including "at_end"), which must  
  be the same position as skip_to() would have moved to.

or

• return false, having moved to an "indeterminate" position, such that a subsequent  
  call to next() or skip_to() will move to the next matching position after did.

Generally, this method should act like skip_to() and return true if that can be done at  
little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is,  
and false if it isn’t.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the  
single subdatabase relevant to this posting source. See the init() method for details.

Parameters

<table>
<thead>
<tr>
<th>did</th>
<th>The document id to check.</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_wt</td>
<td>The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).</td>
</tr>
</tbody>
</table>

Reimplemented from Xapian::ValuePostingSource.

7.14.2.2 DecreasingValueWeightPostingSource& Xapian::-
         DecreasingValueWeightPostingSource::clone ( ) const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie,  
the clone does not need to be in the same iteration position as the original: the matcher  
will always call init() on the clone before attempting to move the iterator, or read the  
information about the current position of the iterator.
This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.3 std::string Xapian::DecreasingValueWeightPostingSource::get_description() const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.4 double Xapian::DecreasingValueWeightPostingSource::get_weight() const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.5 void Xapian::DecreasingValueWeightPostingSource::init(const Xapian::Database & db) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, init() will therefore be called multiple times, and must handle this by using the database passed in the most recent call.
The database which the PostingSource should iterate through.

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.

Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using clone()), and each PostingSource will be passed one of the sub-databases as the db parameter here. The db parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.6 std::string Xapian::DecreasingValueWeightPostingSource::name() const
[virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "-FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.7 void Xapian::DecreasingValueWeightPostingSource::next( double min_wt )
[virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).
7.14.2.8 std::string Xapian::DecreasingValueWeightPostingSource::serialise () const [virtual]

Serialise object parameters into a string.
The serialised parameters should represent the configuration of the posting source, but
need not (indeed, should not) represent the current iteration state.
If you don’t want to support the remote backend, you can use the default implementation
which simply throws Xapian::UnimplementedError.
Reimplemented from Xapian::ValueWeightPostingSource.

7.14.2.9 void Xapian::DecreasingValueWeightPostingSource::skip_to ( Xapian::docid did, double min_wt ) [virtual]

Advance to the specified docid.
If the specified docid isn’t in the list, position ourselves on the first document after it (or
at_end() if no greater docids are present).
If the current position is already the specified docid, this method will leave the position
unmodified.
If the specified docid is earlier than the current position, the behaviour is unspecified.
A sensible behaviour would be to leave the current position unmodified, but it is also
reasonable to move to the specified docid.
The default implementation calls next() repeatedly, which works but skip_to() can often
be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.
Note: in the case of a multi-database search, the docid specified is the docid in the
single subdatabase relevant to this posting source. See the init() method for details.

Parameters

<table>
<thead>
<tr>
<th>did</th>
<th>The document id to advance to.</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_wt</td>
<td>The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).</td>
</tr>
</tbody>
</table>

Reimplemented from Xapian::ValuePostingSource.

7.14.2.10 DecreasingValueWeightPostingSource* Xapian::DecreasingValueWeightPostingSource::unserialise ( const std::string & s ) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete".
If you want to handle the deletion in a special way (for example when wrapping the
Xapian API for use from another language) then you can define a static operator
delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>A serialised instance of this PostingSource subclass.</td>
</tr>
</tbody>
</table>

Reimplemented from Xapian::ValueWeightPostingSource.

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

- xapian/postingsource.h

7.15 Xapian::DocNotFoundError Class Reference

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

Inheritance diagram for Xapian::DocNotFoundError:

```
Xapian::Error
    ▼
   |   
Xapian::RuntimeError
    ▼
   |   
Xapian::DocNotFoundError
```

Public Member Functions

- **DocNotFoundError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- **DocNotFoundError** (const std::string &msg_, int errno_)

  Construct from message and errno value.
7.15.1 Detailed Description

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

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Xapian::DocNotFoundError::DocNotFoundError ( const std::string & msg_, const std::string & context_=std::string(), int errno_=0 ) [inline, explicit] 

General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>msg_</th>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

7.15.2.2 Xapian::DocNotFoundError::DocNotFoundError ( const std::string & msg_, int errno_ ) [inline]

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>msg_</th>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.16 Xapian::Document Class Reference

A handle representing a document in a Xapian database.

Public Member Functions

- Document (const Document &other)
  
  Copying is allowed.
- void operator= (const Document &other)
  
  Assignment is allowed.
- Document ()
  
  Make a new empty Document.
• ~Document ()
  Destructor.
• std::string get_value (Xapian::valueno slot) const
  Get value by number.
• void add_value (Xapian::valueno slot, const std::string &value)
  Add a new value.
• void remove_value (Xapian::valueno slot)
  Remove any value with the given number.
• void clear_values ()
  Remove all values associated with the document.
• std::string get_data () const
  Get data stored in the document.
• void set_data (const std::string &data)
  Set data stored in the document.
• void add_posting (const std::string &tname, Xapian::termpos tpos, Xapian::termcount wdfinc=1)
  Add an occurrence of a term at a particular position.
• void add_term (const std::string &tname, Xapian::termcount wdfinc=1)
  Add a term to the document, without positional information.
• void add_boolean_term (const std::string &term)
  Add a boolean filter term to the document.
• void remove_posting (const std::string &tname, Xapian::termpos tpos, Xapian::termcount wdfdec=1)
  Remove a posting of a term from the document.
• void remove_term (const std::string &tname)
  Remove a term and all postings associated with it.
• void clear_terms ()
  Remove all terms (and postings) from the document.
• Xapian::termcount termlist_count () const
  The length of the termlist - i.e.
• TermIterator termlist_begin () const
  Iterator for the terms in this document.
• TermIterator termlist_end () const
  Equivalent end iterator for termlist_begin().
• Xapian::termcount values_count () const
  Count the values in this document.
• ValueIterator values_begin () const
  Iterator for the values in this document.
• ValueIterator values_end () const
  Equivalent end iterator for values_begin().
• docid get_docid () const
  Get the document id which is associated with this document (if any).
• std::string serialise () const
  Serialise document into a string.
• std::string get_description () const
  Return a string describing this object.
Static Public Member Functions

- static Document unserialise (const std::string &s)
  
  Unserialise a document from a string produced by serialise().

### 7.16.1 Detailed Description

A handle representing a document in a Xapian database.

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

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

We recommend you avoid designs where this behaviour is an issue, but if you need a way to make a non-lazy version of a **Document** object, you can do this like so:

```cpp
doc = Xapian::Document::unserialise(doc.serialise());
```

### 7.16.2 Constructor & Destructor Documentation

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

Copying is allowed.

The internals are reference counted, so copying is cheap.

**Parameters**

| other | The object to copy. |

### 7.16.3 Member Function Documentation

#### 7.16.3.1 void Xapian::Document::add_boolean_term ( const std::string & term )

[inline]

Add a boolean filter term to the document.

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

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

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

This method was added in Xapian 1.0.18.
Parameters

<table>
<thead>
<tr>
<th>term</th>
<th>The term to add.</th>
</tr>
</thead>
</table>

7.16.3.2 void Xapian::Document::add_posting ( const std::string & tname, Xapian::termpos tpos, Xapian::termcount wdfinc = 1 )

Add an occurrence of a term at a particular position.

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

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

Parameters

| tname | The name of the term. |
| tpos  | The position of the term. |
| wdfinc | The increment that will be applied to the wdf for this term. |

7.16.3.3 void Xapian::Document::add_term ( const std::string & tname, Xapian::termcount wdfinc = 1 )

Add a term to the document, without positional information.

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

Parameters

| tname | The name of the term. |
| wdfinc | The increment that will be applied to the wdf for this term (default: 1). |

7.16.3.4 void Xapian::Document::add_value ( Xapian::valueno slot, const std::string & value )

Add a new value.

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

Parameters

| slot | The value slot to add the value in. |
| value | The value to set. |
7.16.3.5 std::string Xapian::Document::get_data ( ) const

Get data stored in the document.

This is potentially a relatively expensive operation, and shouldn't normally be used during the match (e.g. in a PostingSource or match decider functor. Put data for use by match deciders in a value instead.

7.16.3.6 docid Xapian::Document::get_docid ( ) const

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

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

Returns

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

7.16.3.7 std::string Xapian::Document::get_value ( Xapian::valueno slot ) const

Get value by number.

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

Parameters

| slot | The number of the value. |

7.16.3.8 void Xapian::Document::operator= ( const Document & other )

Assignment is allowed.

The internals are reference counted, so assignment is cheap.

Parameters

| other | The object to copy. |

7.16.3.9 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tname</code></td>
<td>The name of the term.</td>
</tr>
<tr>
<td><code>tpos</code></td>
<td>The position of the term.</td>
</tr>
<tr>
<td><code>wdfdec</code></td>
<td>The decrement that will be applied to the wdf when removing this posting. The wdf will not go below the value of 0.</td>
</tr>
</tbody>
</table>

### 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.

#### 7.16.3.10 void Xapian::Document::remove_term ( const std::string & `tname` )

Remove a term and all postings associated with it.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tname</code></td>
<td>The name of the term.</td>
</tr>
</tbody>
</table>

### Exceptions

- `Xapian::InvalidArgumentError` will be thrown if the term is not in the document.

#### 7.16.3.11 std::string Xapian::Document::serialise ( ) const

Serialise document into a string.

The document representation may change between Xapian releases: even between minor versions. However, it is guaranteed not to change if the remote database protocol has not changed between releases.

#### 7.16.3.12 void Xapian::Document::set_data ( const std::string & `data` )

Set data stored in the document.

Xapian treats the data as an opaque blob. It may try to compress it, but other than that it will just store it and return it when requested.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>data</code></td>
<td>The data to store.</td>
</tr>
</tbody>
</table>
7.16.3.13 Xapian::termcount Xapian::Document::termlist_count ( ) const

The length of the termlist - i.e.
the number of different terms which index this document.
The documentation for this class was generated from the following file:

- xapian/document.h

7.17 Xapian::Enquire Class Reference

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

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)
  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 **add_matchspy** (MatchSpy *spy)
  Add a matchspy.
- void **clear_matchspies** ()
  Remove all the matchspies.
- void **set_weighting_scheme** (const Weight &weight_)
  Set the weighting scheme to use for queries.
- void **setCollapseKey** (Xapian::valueno collapse_key, Xapian::doccount collapse_max=1)
  Set the collapse key to use for queries.
- void **setDocidOrder** (docid_order order)
  Set the direction in which documents are ordered by document id in the returned MSet.
- void **setCutoff** (int percent_cutoff, double weight_cutoff=0)
  Set the percentage and/or weight cutoffs.
- void **setSortByRelevance** ()
  Set the sorting to be by relevance only.

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
• void set_sort_by_value (Xapian::valueno sort_key, bool reverse)
  Set the sorting to be by value only.
• void set_sort_by_key (Xapian::KeyMaker *sorter, bool reverse)
  Set the sorting to be by key generated from values only.
• void set_sort_by_value_then_relevance (Xapian::valueno sort_key, bool reverse)
  Set the sorting to be by value, then by relevance for documents with the same value.
• void set_sort_by_key_then_relevance (Xapian::KeyMaker *sorter, bool reverse)
  Set the sorting to be by keys generated from values, then by relevance for documents with identical keys.
• void set_sort_by_relevance_then_value (Xapian::valueno sort_key, bool reverse)
  Set the sorting to be by relevance then value.
• void set_sort_by_relevance_then_key (Xapian::KeyMaker *sorter, bool reverse)
  Set the sorting to be by relevance, then by keys generated from values.
• 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.
• ESet get_eset (Xapian::termcount maxitems, const RSet &omrset, int flags, double k, const Xapian::ExpandDecider *edecider, double min_wt) 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 did) 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()
• std::string get_description () const
  Return a string describing this object.

• 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, const RSet &omrset, const MatchDecider *mdecider=0) const
  Get (a portion of) the match set for the current query.
7.17 Xapian::Enquire Class Reference

7.17.1 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::InvalidArgument will be thrown if an invalid argument is supplied, for example, an unknown database type.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Xapian::Enquire::Enquire ( const Database & database ) [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

<table>
<thead>
<tr>
<th>database</th>
<th>Specification of the database or databases to use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>errorhandler</td>
<td>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.</td>
</tr>
</tbody>
</table>

Exceptions

Xapian::InvalidArgument will be thrown if an empty Database object is supplied.

7.17.3 Member Function Documentation

7.17.3.1 void Xapian::Enquire::add_matchspy ( MatchSpy * spy )

Add a matchspy.

This matchspy will be called with some of the documents which match the query, during the match process. Exactly which of the matching documents are passed to it depends on exactly when certain optimisations occur during the match process, but it can be

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
controlled to some extent by setting the `checkatleast` parameter to `get_mset()`.

In particular, if there are enough matching documents, at least the number specified by `checkatleast` will be passed to the matchspy. This means that you can force the matchspy to be shown all matching documents by setting `checkatleast` to the number of documents in the database.

Parameters

| spy               | The MatchSpy subclass to add. The caller must ensure that this remains valid while the Enquire object remains active, or until `clear_matchspies()` is called. |

### 7.17.3.2 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: |
| `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. |

### 7.17.3.3 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.
7.17 Xapian::Enquire Class Reference

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxitems</td>
<td>the maximum number of items to return.</td>
</tr>
<tr>
<td>omrset</td>
<td>the relevance set to use when performing the expand operation.</td>
</tr>
<tr>
<td>edecider</td>
<td>a decision functor to use to decide whether a given term should be put in the ESet</td>
</tr>
</tbody>
</table>

Returns

An ESet object containing the results of the expand.

Exceptions

Xapian::InvalidArgumentError  See class documentation.

7.17.3.4  ESet Xapian::Enquire::get_eset ( Xapian::termcount maxitems, const RSet & omrset, int flags, double k, const Xapian::ExpandDecider * edecider, double min_wt ) const

Get the expand set for the given rset.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxitems</td>
<td>the maximum number of items to return.</td>
</tr>
<tr>
<td>omrset</td>
<td>the relevance set to use when performing the expand operation.</td>
</tr>
<tr>
<td>flags</td>
<td>zero or more of these values</td>
</tr>
<tr>
<td></td>
<td>• Xapian::Enquire::INCLUDE_QUERY_TERMS query terms may be returned from expand</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td>k</td>
<td>the parameter k in the query expansion algorithm (default is 1.0)</td>
</tr>
<tr>
<td>edecider</td>
<td>a decision functor to use to decide whether a given term should be put in the ESet</td>
</tr>
<tr>
<td>min_wt</td>
<td>the minimum weight for included terms</td>
</tr>
</tbody>
</table>

Returns

An ESet object containing the results of the expand.

Exceptions

Xapian::InvalidArgumentError  See class documentation.
7.17.3.5 TermIterator Xapian::Enquire::get_matching_terms_begin (Xapian::docid did) const

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

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

It is possible for the document to have been removed from the database between the time it is returned in an MSet, and the time that this call is made. If possible, you should specify an MSetIterator instead of a Xapian::docid, since this will enable database back-ends 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.

7.17.3.6 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.
7.17 Xapian::Enquire Class Reference

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::InvalidArgumentError</td>
<td>See class documentation.</td>
</tr>
<tr>
<td>Xapian::DocNotFoundError</td>
<td>The document specified could not be found in the database.</td>
</tr>
</tbody>
</table>

7.17.3.7 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>first</td>
<td>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.</td>
</tr>
<tr>
<td>maxitems</td>
<td>the maximum number of items to return. If you want all matches, then you can pass the result of calling get_doccount() on the Database object (though if you are doing this so you can filter results, you are likely to get much better performance by using Xapian's match-time filtering features instead). You can pass 0 for maxitems which will give you an empty MSet with valid statistics (such as get_matches_estimated()) calculated without looking at any postings, which is very quick, but means the estimates may be more approximate and the bounds may be much looser.</td>
</tr>
<tr>
<td>checkatleast</td>
<td>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.</td>
</tr>
<tr>
<td>omrset</td>
<td>the relevance set to use when performing the query.</td>
</tr>
<tr>
<td>mdecider</td>
<td>a decision functor to use to decide whether a given document should be put in the MSet.</td>
</tr>
</tbody>
</table>

Returns

A Xapian::MSet object containing the results of the query.

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::InvalidArgumentError</td>
<td>See class documentation.</td>
</tr>
</tbody>
</table>
Get (a portion of) the match set for the current query.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>first</code></td>
<td>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.</td>
</tr>
<tr>
<td><code>maxitems</code></td>
<td>the maximum number of items to return. If you want all matches, then you can pass the result of calling <code>get_doccount()</code> on the <code>Database</code> object (though if you are doing this so you can filter results, you are likely to get much better performance by using Xapian's match-time filtering features instead). You can pass 0 for <code>maxitems</code> which will give you an empty <code>MSet</code> with valid statistics (such as <code>get_matches_estimated()</code>) calculated without looking at any postings, which is very quick, but means the estimates may be more approximate and the bounds may be much looser.</td>
</tr>
<tr>
<td><code>checkatleast</code></td>
<td>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 <code>checkatleast</code> forces it to consider at least this many matches and so allows for reliable paging links.</td>
</tr>
<tr>
<td><code>omrset</code></td>
<td>the relevance set to use when performing the query.</td>
</tr>
<tr>
<td><code>mdecider</code></td>
<td>a decision functor to use to decide whether a given document should be put in the <code>MSet</code>.</td>
</tr>
</tbody>
</table>

Returns

A Xapian::MSet object containing the results of the query.

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::InvalidArgumentError</td>
<td>See class documentation.</td>
</tr>
</tbody>
</table>

**7.17.3.9**

const Xapian::Query& Xapian::Enquire::get_query ( ) const

Get the query which has been set.

This is only valid after `set_query()` has been called.

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::InvalidArgumentError</td>
<td>will be thrown if query has not yet been set.</td>
</tr>
</tbody>
</table>
Set the collapse key to use for queries.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>collapse_key</code></td>
<td>value number to collapse on - at most one MSet entry with each particular value will be returned (default is Xapian::BAD_VALUENO which means no collapsing).</td>
</tr>
<tr>
<td><code>collapse_max</code></td>
<td>Max number of items with the same key to leave after collapsing (default 1).</td>
</tr>
</tbody>
</table>

The MSet returned by `get_mset()` will have only the "best" (at most) `collapse_max` entries with each particular value of `collapse_key` ("best" being highest ranked - i.e. 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 by collapsing with `collapse_max = 1` (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 `getCollapseCount()` 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.

Set the percentage and/or weight cutoffs.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>percent_cutoff</code></td>
<td>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 <code>set_cutoff(100)</code>. (default 0 =&gt; no percentage cut-off).</td>
</tr>
<tr>
<td><code>weight_cutoff</code></td>
<td>Minimum weight for a document to be returned. If a document has a lower score that 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 =&gt; no weight cut-off).</td>
</tr>
</tbody>
</table>
7.17.3.12 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

<table>
<thead>
<tr>
<th>order</th>
<th>This can be:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Xapian::Enquire::ASCENDING docids sort</td>
</tr>
<tr>
<td></td>
<td>in ascending order (default)</td>
</tr>
<tr>
<td></td>
<td>• Xapian::Enquire::DESCENDING docids sort</td>
</tr>
<tr>
<td></td>
<td>in descending order</td>
</tr>
<tr>
<td></td>
<td>• Xapian::Enquire::DONT_CARE docids sort</td>
</tr>
<tr>
<td></td>
<td>in whatever order is most efficient for</td>
</tr>
<tr>
<td></td>
<td>the backend</td>
</tr>
</tbody>
</table>

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 an efficient way to perform "sort by date, newest first", and with set_docid_order(Xapian::Enquire::ASCENDING) a very efficient way to perform "sort by date, oldest first".

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

Set the query to run.

Parameters

<table>
<thead>
<tr>
<th>query</th>
<th>the new query to run.</th>
</tr>
</thead>
<tbody>
<tr>
<td>qlen</td>
<td>the query length to use in weight calculations - by default the sum of the wqf of all terms is used.</td>
</tr>
</tbody>
</table>

7.17.3.14 void Xapian::Enquire::set_sort_by_key ( Xapian::KeyMaker * sorter, bool reverse )

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

Parameters

<table>
<thead>
<tr>
<th>sorter</th>
<th>The functor to use for generating keys.</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
<td>If true, reverses the sort order.</td>
</tr>
</tbody>
</table>
7.17.3.15  void Xapian::Enquire::set_sort_by_key_then_relevance (  
        Xapian::KeyMaker * sorter, bool reverse )

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

Parameters

<table>
<thead>
<tr>
<th>sorter</th>
<th>The functor to use for generating keys.</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
<td>If true, reverses the sort order.</td>
</tr>
</tbody>
</table>

7.17.3.16  void Xapian::Enquire::set_sort_by_relevance (  )

Set the sorting to be by relevance only.  
This is the default.

7.17.3.17  void Xapian::Enquire::set_sort_by_relevance_then_key (  
        Xapian::KeyMaker * sorter, bool reverse )

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

<table>
<thead>
<tr>
<th>sorter</th>
<th>The functor to use for generating keys.</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverse</td>
<td>If true, reverses the sort order of the generated keys.</td>
</tr>
</tbody>
</table>

7.17.3.18  void Xapian::Enquire::set_sort_by_relevance_then_value (  
        Xapian::valueno sort_key, bool reverse )

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort_key</td>
<td>value number to sort on.</td>
</tr>
<tr>
<td>reverse</td>
<td>If true, reverses the sort order of sort_key.</td>
</tr>
</tbody>
</table>

7.17.3.19 void Xapian::Enquire::set_sort_by_value ( Xapian::valueno sort_key, bool reverse )

Set the sorting to be by value only.

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort_key</td>
<td>value number to sort on.</td>
</tr>
<tr>
<td>reverse</td>
<td>If true, reverses the sort order.</td>
</tr>
</tbody>
</table>

7.17.3.20 void Xapian::Enquire::set_sort_by_value_then_relevance ( Xapian::valueno sort_key, bool reverse )

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

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

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sort_key</td>
<td>value number to sort on.</td>
</tr>
<tr>
<td>reverse</td>
<td>If true, reverses the sort order.</td>
</tr>
</tbody>
</table>

7.17.3.21 void Xapian::Enquire::set_weighting_scheme ( const Weight & weight )

Set the weighting scheme to use for queries.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>the new weighting scheme. If no weighting scheme is specified, the default is BM25 with the default parameters.</td>
</tr>
</tbody>
</table>

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

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
All exceptions thrown by Xapian are subclasses of Xapian::Error.

Inheritance diagram for Xapian::Error:

Public Member Functions

- const char * get_type () const
  
  The type of this error (e.g. "DocNotFoundError").

- const std::string & get_msg () const
  
  Message giving details of the error, intended for human consumption.

- const std::string & get_context () const
  
  Optional context information.

- const char * get_error_string () const
  
  Returns any system error string associated with this exception.

- std::string get_description () const
  
  Return a string describing this object.

7.18.1 Detailed Description

All exceptions thrown by Xapian are subclasses of Xapian::Error.

This class can not be instantiated directly - instead a subclass should be used.
7.18.2 Member Function Documentation

7.18.2.1 const std::string& Xapian::Error::get_context ( ) const [inline]

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

7.18.2.2 const char∗ Xapian::Error::get_error_string ( ) const

Returns any system error string associated with this exception.
The system error string may come from errno, h_errno (on UNIX), or GetLastError() (on MS Windows). If there is no associated system error string, NULL is returned.
The documentation for this class was generated from the following file:

- xapian/error.h

7.19 Xapian::ErrorHandler Class Reference

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

Public Member Functions

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

7.19.1 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).
7.19.2 Member Function Documentation

7.19.2.1 void Xapian::ErrorHandler::operator() (Xapian::Error &error)

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

Parameters

| error  | The Xapian::Error object under consideration. |

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

- xapian/errorhandler.h

7.20 Xapian::ESet Class Reference

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

Public Member Functions

- **ESet ()**
  
  Construct an empty ESet.

- **∼ESet ()**

  Destructor.

- **ESet (const ESet &other)**

  Copying is allowed (and is cheap).

- **void operator= (const ESet &other)**

  Assignment is allowed (and is cheap).

- **Xapian::termcount get_ebound () const**

  A lower bound on the number of terms which are in the full set of results of the expand.

- **Xapian::termcount size () const**

  The number of terms in this E-Set.

- **Xapian::termcount max_size () const**

  Required to allow use as an STL container.

- **bool empty () const**

  Test if this E-Set is empty.

- **void swap (ESet &other)**

  Swap the E-Set we point to with another.

- **ESetIterator begin () const**

  Iterator for the terms in this E-Set.
• `ESetIterator end () const`
  `End iterator corresponding to begin()`
• `ESetIterator back () const`
  `Iterator pointing to the last element of this E-Set.`
• `ESetIterator operator[] (Xapian::termcount i) const`
  `This returns the term at position i in this E-Set.`
• `std::string get_description () const`
  `Return a string describing this object.`

### 7.20.1 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()`.

### 7.20.2 Member Function Documentation

#### 7.20.2.1 `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()`

#### 7.20.2.2 `Xapian::termcount Xapian::ESet::max_size ( ) const [inline]`

Required to allow use as an STL container.

#### 7.20.2.3 `ESetIterator Xapian::ESet::operator[] (Xapian::termcount i) const`

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

**Parameters**

| i  | The index into the ESet. |

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

• `xapian/enquire.h`

### 7.21 Xapian::ESetIterator Class Reference

Iterate through terms in the ESet.
Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  
  Allow use as an STL iterator.
- typedef std::string value_type
  
  Allow use as an STL iterator.
- typedef Xapian::termcount_diff difference_type
  
  Allow use as an STL iterator.
- typedef std::string * pointer
  
  Allow use as an STL iterator.
- typedef std::string & reference
  
  Allow use as an STL iterator.

Public Member Functions

- ESetIterator ()
  Create an uninitialised iterator; this cannot be used, but is convenient syntactically.
- ESetIterator (const ESetIterator &other)
  Copying is allowed (and is cheap).
- void operator= (const ESetIterator &other)
  Assignment is allowed (and is cheap).
- ESetIterator & operator++ ()
  Advance the iterator.
- ESetIterator operator++ (int)
  Advance the iterator (postfix variant).
- ESetIterator & operator-- ()
  Decrement the iterator.
- ESetIterator operator-- (int)
  Decrement the iterator (postfix variant).
- const std::string & operator* () const
  Get the term for the current position.
- double get_weight () const
  Get the weight of the term at the current position.
- std::string get_description () const
  Return a string describing this object.

Friends

- bool operator== (const ESetIterator &a, const ESetIterator &b)
  Equality test for ESetIterator objects.
- bool operator!= (const ESetIterator &a, const ESetIterator &b)
  Inequality test for ESetIterator objects.
7.21.1 Detailed Description

Iterate through terms in the ESet.

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

- xapian/enquire.h

7.22 Xapian::ExpandDecider Class Reference

Virtual base class for expand decider functor.

Inheritance diagram for Xapian::ExpandDecider:

Public Member Functions

- virtual bool operator() (const std::string &term) const =0
  
  Do we want this term in the ESet?

- virtual ~ExpandDecider ()
  
  Virtual destructor, because we have virtual methods.

7.22.1 Detailed Description

Virtual base class for expand decider functor.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 virtual Xapian::ExpandDecider::~ExpandDecider () [virtual]

Virtual destructor, because we have virtual methods.
7.23 Xapian::ExpandDeciderAnd Class Reference

7.22.3 Member Function Documentation

7.22.3.1 virtual bool Xapian::ExpandDecider::operator() ( const std::string & term ) const

[pure virtual]

Do we want this term in the ESet?

Parameters

| term | The term to test. |

Implemented in Xapian::ExpandDeciderFilterPrefix, Xapian::ExpandDeciderFilterTerms, and Xapian::ExpandDeciderAnd.

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

- xapian/expanddecider.h

7.23 Xapian::ExpandDeciderAnd Class Reference

ExpandDecider subclass which rejects terms using two ExpandDeciders.

Inheritance diagram for Xapian::ExpandDeciderAnd:

```
Xapian::ExpandDecider
    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?

7.23.1 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.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Xapian::ExpandDeciderAnd::ExpandDeciderAnd ( const ExpandDecider & first_, const ExpandDecider & second_ ) [inline]

Terms will be checked with first_, and if accepted, then checked with second_.

Parameters

| first_ | First ExpandDecider object to test with. |
| second_ | ExpandDecider object to test with if first_ accepts. |

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

Compatibility method.

Parameters

| first_ | First ExpandDecider object to test with. |
| second_ | ExpandDecider object to test with if first_ accepts. |

7.23.3 Member Function Documentation

7.23.3.1 virtual bool Xapian::ExpandDeciderAnd::operator() ( const std::string & term ) const [virtual]

Do we want this term in the ESet?

Parameters

| term | The term to test. |

Implements Xapian::ExpandDecider.

The documentation for this class was generated from the following file:
7.24 Xapian::ExpandDeciderFilterPrefix Class Reference

ExpandDecider subclass which restrict terms to a particular prefix.

Inheritance diagram for Xapian::ExpandDeciderFilterPrefix:

```
Xapian::ExpandDeciderFilterPrefix
```

Public Member Functions

- **ExpandDeciderFilterPrefix** (const std::string &prefix_) 
  
  The parameter specify the prefix of terms to be retained.

- virtual bool **operator()** (const std::string &term) const
  
  Do we want this term in the ESet?

7.24.1 Detailed Description

ExpandDecider subclass which restrict terms to a particular prefix.

ExpandDeciderFilterPrefix provides an easy way to choose terms with a particular prefix when generating an ESet.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 Xapian::ExpandDeciderFilterPrefix::ExpandDeciderFilterPrefix ( const std::string & prefix_ ) [inline]

The parameter specify the prefix of terms to be retained.
Parameters

| prefix_ | restrict terms to the particular prefix_ |

7.24.3 Member Function Documentation

7.24.3.1 virtual bool Xapian::ExpandDeciderFilterPrefix::operator() (const std::string & term) const [virtual]

Do we want this term in the ESet?

Parameters

| term | The term to test. |

Implements Xapian::ExpandDecider.

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

- xapian/expanddecider.h

7.25 Xapian::ExpandDeciderFilterTerms Class Reference

ExpandDecider subclass which rejects terms in a specified list.

Inheritance diagram for Xapian::ExpandDeciderFilterTerms:

```
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?

7.25.1 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.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 template <class Iterator> Xapian::ExpandDeciderFilterTerms::ExpandDeciderFilterTerms ( Iterator reject_begin, Iterator reject_end ) [inline]

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

Parameters

| reject_begin | Begin iterator for the list of terms to reject. It can be any input_iterator type which returns std::string or char * (e.g. TermIterator or char **). |
| reject_end   | End iterator for the list of terms to reject. |

7.25.3 Member Function Documentation

7.25.3.1 virtual bool Xapian::ExpandDeciderFilterTerms::operator() ( const std::string & term ) const [virtual]

Do we want this term in the ESet?

Parameters

| term | The term to test. |

Implements Xapian::ExpandDecider.

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

- xapian/expanddecider.h

7.26 Xapian::FeatureUnavailableError Class Reference

Indicates an attempt to use a feature which is unavailable.
Inheritance diagram for Xapian::FeatureUnavailableError:

```
Xapian::Error

Xapian::RuntimeError

Xapian::FeatureUnavailableError
```

Public Member Functions

- **FeatureUnavailableError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.

- **FeatureUnavailableError** (const std::string &msg_, int errno_)
  
  Construct from message and errno value.

7.26.1 Detailed Description

Indicates an attempt to use a feature which is unavailable.

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

7.26.2 Constructor & Destructor Documentation

7.26.2.1 **Xapian::FeatureUnavailableError::FeatureUnavailableError** (const std::string & msg_, const std::string & context_=std::string(), int errno_=0) [inline, explicit]

General purpose constructor.
7.27 Xapian::FieldProcessor Struct Reference

Base class for field processors.

Public Member Functions

- virtual ~FieldProcessor()
  Destructor.
- virtual Xapian::Query operator() (const std::string &str)=0
  Convert a field-prefixed string to a Query object.

7.27.1 Detailed Description

Base class for field processors.
Experimental API - may change.

7.27.2 Member Function Documentation

7.27.2.1 virtual Xapian::Query Xapian::FieldProcessor::operator() (const std::string &str) [pure virtual]

Convert a field-prefixed string to a Query object.
Parameters

| str | The string to convert. |

Returns

- Query object corresponding to str.

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

- xapian/queryparser.h

### 7.28 Xapian::FixedWeightPostingSource Class Reference

A posting source which returns a fixed weight for all documents.

Inheritance diagram for Xapian::FixedWeightPostingSource:

```plaintext
Xapian::FixedWeightPostingSource
  ↓
Xapian::PostingSource
```

Public Member Functions

- **FixedWeightPostingSource**(double wt)
  
  Construct a FixedWeightPostingSource.

- **Xapian::doccount get_termfreq_min**( ) const
  
  A lower bound on the number of documents this object can return.

- **Xapian::doccount get_termfreq_est**( ) const
  
  An estimate of the number of documents this object can return.

- **Xapian::doccount get_termfreq_max**( ) const
  
  An upper bound on the number of documents this object can return.

- **double get_weight**( ) const
  
  Return the weight contribution for the current document.

- **void next**(double min_wt)
7.28 Xapian::FixedWeightPostingSource Class Reference

Advance the current position to the next matching document.

- void skip_to (Xapian::docid min_docid, double min_wt)
  Advance to the specified docid.

- bool check (Xapian::docid min_docid, double min_wt)
  Check if the specified docid occurs.

- bool at_end () const
  Return true if the current position is past the last entry in this list.

- Xapian::docid get_docid () const
  Return the current docid.

- FixedWeightPostingSource * clone () const
  Clone the posting source.

- std::string name () const
  Name of the posting source class.

- std::string serialise () const
  Serialise object parameters into a string.

- FixedWeightPostingSource * unserialise (const std::string &s) const
  Create object given string serialisation returned by serialise().

- void init (const Database &db_)
  Set this PostingSource to the start of the list of postings.

- std::string get_description () const
  Return a string describing this object.

7.28.1 Detailed Description

A posting source which returns a fixed weight for all documents.

This returns entries for all documents in the given database, with a fixed weight (specified by a parameter to the constructor).

7.28.2 Constructor & Destructor Documentation

7.28.2.1 Xapian::FixedWeightPostingSource::FixedWeightPostingSource ( double wt )

Construct a FixedWeightPostingSource.

Parameters:

| wt | The fixed weight to return. |

7.28.3 Member Function Documentation

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.28.3.1 bool Xapian::FixedWeightPostingSource::at_end() const [virtual]

Return true if the current position is past the last entry in this list.
At least one of next(), skip_to() or check() will be called before this method is first called.
Implements Xapian::PostingSource.

7.28.3.2 bool Xapian::FixedWeightPostingSource::check( Xapian::docid did, double min_wt ) [virtual]

Check if the specified docid occurs.
The caller is required to ensure that the specified document id did actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after did.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.
Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn’t.
The default implementation calls skip_to() and always returns true.
Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the init() method for details.

Parameters

<table>
<thead>
<tr>
<th>did</th>
<th>The document id to check.</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_wt</td>
<td>The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).</td>
</tr>
</tbody>
</table>

Reimplemented from Xapian::PostingSource.

7.28.3.3 FixedWeightPostingSource* Xapian::FixedWeightPostingSource::clone() const [virtual]

Clone the posting source.
The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

Reimplemented from Xapian::PostingSource.

7.28.3.4 std::string Xapian::FixedWeightPostingSource::get_description() const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

7.28.3.5 Xapian::docid Xapian::FixedWeightPostingSource::get_docid() const [virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See get_weight() for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the init() method for details.

Implements Xapian::PostingSource.

7.28.3.6 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_est() const [virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.

Implements Xapian::PostingSource.
7.28.3.7 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_max ( ) const [virtual]

An upper bound on the number of documents this object can return.
Xapian will always call init() on a PostingSource before calling this for the first time.
Implements Xapian::PostingSource.

7.28.3.8 Xapian::doccount Xapian::FixedWeightPostingSource::get_termfreq_min ( ) const [virtual]

A lower bound on the number of documents this object can return.
Xapian will always call init() on a PostingSource before calling this for the first time.
Implements Xapian::PostingSource.

7.28.3.9 double Xapian::FixedWeightPostingSource::get_weight ( ) const [virtual]

Return the weight contribution for the current document.
This default implementation always returns 0, for convenience when implementing
"weight-less" PostingSource subclasses.
This method may assume that it will only be called when there is a "current document".
In detail: Xapian will always call init() on a PostingSource before calling this for the first
time. It will also only call this if the PostingSource reports that it is pointing to a valid
document (ie, it will not call it before calling at least one of next(), skip_to() or check(),
and will ensure that the PostingSource is not at the end by calling at_end()).
Reimplemented from Xapian::PostingSource.

7.28.3.10 void Xapian::FixedWeightPostingSource::init ( const Database & db ) [virtual]

Set this PostingSource to the start of the list of postings.
This is called automatically by the matcher prior to each query being processed.
If a PostingSource is used for multiple searches, init() will therefore be called multiple
times, and must handle this by using the database passed in the most recent call.

Parameters

| db | The database which the PostingSource should iterate through. |

Note: the database supplied to this method must not be modified: in particular, the
reopen() method should not be called on it.
Note: in the case of a multi-database search, a separate PostingSource will be used for
each database (the separate PostingSources will be obtained using clone()), and each
PostingSource will be passed one of the sub-databases as the \textit{db} parameter here. The \textit{db} parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Implements Xapian::PostingSource.

7.28.3.11 \texttt{std::string Xapian::FixedWeightPostingSource::name ( ) const}  

[virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "-FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

7.28.3.12 \texttt{void Xapian::FixedWeightPostingSource::next ( double min_wt )}  

[virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

Parameters

\begin{table}[h]
\centering
\begin{tabular}{|l|p{10cm}|}
\hline
\textit{min_wt} & The minimum weight contribution that is needed (this is just a hint which subclasses may ignore). \\
\hline
\end{tabular}
\end{table}

Implements Xapian::PostingSource.

7.28.3.13 \texttt{std::string Xapian::FixedWeightPostingSource::serialise ( ) const}  

[virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.
If you don’t want to support the remote backend, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Reimplemented from `Xapian::PostingSource`.

### 7.28.3.14 `void Xapian::FixedWeightPostingSource::skip_to` (Xapian::docid `did`, double `min_wt`) [virtual]

Advance to the specified docid.

- If the specified docid isn't in the list, position ourselves on the first document after it (or `at_end()` if no greater docids are present).
- If the current position is already the specified docid, this method will leave the position unmodified.
- If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls `next()` repeatedly, which works but `skip_to()` can often be implemented much more efficiently.

`Xapian` will always call `init()` on a `PostingSource` before calling this for the first time.

**Note:** in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the `init()` method for details.

**Parameters**

| `did` | The document id to advance to. |
| `min_wt` | The minimum weight contribution that is needed (this is just a hint which subclasses may ignore). |

Reimplemented from `Xapian::PostingSource`.

### 7.28.3.15 `FixedWeightPostingSource::unserialise` (const std::string & `s`) const [virtual]

Create object given string serialisation returned by `serialise()`.

**Note:** that the returned object will be deallocated by `Xapian` after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the `Xapian` API for use from another language) then you can define a static `operator delete` method in your subclass as shown here: [http://trac.xapian.org/ticket/554#comment:1](http://trac.xapian.org/ticket/554#comment:1)

If you don’t want to support the remote backend, you can use the default implementation which simply throws `Xapian::UnimplementedError`. 
Calculate the great-circle distance between two coordinates on a sphere.

Inheritance diagram for Xapian::GreatCircleMetric:

```
Xapian::LatLongMetric

Xapian::GreatCircleMetric
```

### Public Member Functions

- **GreatCircleMetric ()**
  - Construct a `GreatCircleMetric`.
- **GreatCircleMetric (double radius_)**
  - Construct a `GreatCircleMetric` using a specified radius.
- **double pointwise_distance (const LatLongCoord &a, const LatLongCoord &b) const**
  - Return the great-circle distance between points on the sphere.
- **LatLongMetric * clone () const**
  - Clone the metric.
- **std::string name () const**
  - Return the full name of the metric.
- **std::string serialise () const**
  - Serialise object parameters into a string.
- **LatLongMetric * unserialise (const std::string &s) const**
  - Create object given string serialisation returned by `serialise()`.

---

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.29.1 Detailed Description

Calculate the great-circle distance between two coordinates on a sphere.

Experimental - see http://xapian.org/docs/deprecation#experimental-features

This uses the haversine formula to calculate the distance. Note that this formula is subject to inaccuracy due to numerical errors for coordinates on the opposite side of the sphere.

See http://en.wikipedia.org/wiki/Haversine_formula

7.29.2 Constructor & Destructor Documentation

7.29.2.1 Xapian::GreatCircleMetric::GreatCircleMetric ( )

Construct a GreatCircleMetric.

The (quadratic mean) radius of the Earth will be used by this calculator.

7.29.2.2 Xapian::GreatCircleMetric::GreatCircleMetric ( double radius_ ) [explicit]

Construct a GreatCircleMetric using a specified radius.

This is useful for data sets in which the points are not on Earth (eg, a database of features on Mars).

Parameters

radius_ The radius of the sphere to use, in metres.

7.29.3 Member Function Documentation

7.29.3.1 LatLongMetric Xapian::GreatCircleMetric::clone ( ) const [virtual]

Clone the metric.

Implements Xapian::LatLongMetric.

7.29.3.2 std::string Xapian::GreatCircleMetric::name ( ) const [virtual]

Return the full name of the metric.

This is used when serialising and unserialising metrics; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a LatLongMetric subclass called "FooLatLongMetric" in the "Xapian" namespace the result of this call should be "Xapian::FooLatLongMetric".

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.30 Xapian::InternalError Class Reference

`Xapian::InternalError` indicates a runtime problem of some sort.

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
Inheritance diagram for Xapian::InternalError:

```
Xapian::Error
   Xapian::RuntimeError
      Xapian::InternalError
```

Public Member Functions

- **InternalError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.

- **InternalError** (const std::string &msg_, int errno_)
  
  Construct from message and errno value.

7.30.1 Detailed Description

**InternalError** indicates a runtime problem of some sort.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 Xapian::InternalError::InternalError ( const std::string & msg_, const std::string & context_=std::string(), int errno_=0 ) [inline, explicit]

General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>
7.30.2.2 Xapian::InternalError::InternalError (const std::string &msg_, int errno_)
[inline]

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>msg_</th>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.31 Xapian::InvalidArgumentError Class Reference

InvalidArgumentError indicates an invalid parameter value was passed to the API.

Inheritance diagram for Xapian::InvalidArgumentError:

```
      Xapian::Error
       /
     /   
Xapian::LogicError
       /
      /    
Xapian::InvalidArgumentError
```

Public Member Functions

- **InvalidArgumentError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.

- **InvalidArgumentError** (const std::string &msg_, int errno_)
  
  Construct from message and errno value.
7.31.1 Detailed Description

InvalidArgumentError indicates an invalid parameter value was passed to the API.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 Xapian::InvalidArgumentError::InvalidArgumentError ( const std::string & msg_, const std::string & context_ = std::string(), int errno_ = 0 ) [inline, explicit]

General purpose constructor.

Parameters

- `msg_` Message giving details of the error, intended for human consumption.
- `context_` Optional context information for this error.
- `errno_` Optional errno value associated with this error.

7.31.2.2 Xapian::InvalidArgumentError::InvalidArgumentError ( const std::string & msg_, int errno_ ) [inline]

Construct from message and errno value.

Parameters

- `msg_` Message giving details of the error, intended for human consumption.
- `errno_` Optional errno value associated with this error.

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

- xapian/error.h

7.32 Xapian::InvalidOperationError Class Reference

InvalidOperationError indicates the API was used in an invalid way.
### Public Member Functions

- `InvalidOperationError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)`
  General purpose constructor.
- `InvalidOperationError (const std::string &msg_, int errno_)` Construct from message and errno value.

#### 7.32.1 Detailed Description

`InvalidOperationError` indicates the API was used in an invalid way.

#### 7.32.2 Constructor & Destructor Documentation

7.32.2.1 `Xapian::InvalidOperationError::InvalidOperationError ( const std::string & msg_, const std::string & context_ = std::string(), int errno_ = 0 ) [inline, explicit]`

General purpose constructor.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg_</code></td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td><code>context_</code></td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td><code>errno_</code></td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>
7.32.2.2 Xapian::InvalidOperationError::InvalidOperationError (const std::string & msg_, int errno_) [inline]

Construct from message and errno value.

Parameters

- `msg_`: Message giving details of the error, intended for human consumption.
- `errno_`: Optional errno value associated with this error.

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

- xapian/error.h

7.33 Xapian::KeyMaker Class Reference

Virtual base class for key making functors.

Inheritance diagram for Xapian::KeyMaker:

```
Xapian::KeyMaker
```

```
Xapian::LatLongDistanceKeyMaker
Xapian::MultiValueKeyMaker
```

Public Member Functions

- virtual std::string operator() (const Xapian::Document &doc) const =0

  *Build a key string for a Document.*

- virtual ~KeyMaker ()

  *Virtual destructor, because we have virtual methods.*

7.33.1 Detailed Description

Virtual base class for key making functors.
7.34 Xapian::LatLongCoord Struct Reference

7.33.2 Constructor & Destructor Documentation

7.33.2.1 virtual Xapian::KeyMaker::~KeyMaker() [virtual]

Virtual destructor, because we have virtual methods.

7.33.3 Member Function Documentation

7.33.3.1 virtual std::string Xapian::KeyMaker::operator() (const Xapian::Document &doc) const [pure virtual]

Build a key string for a Document.
These keys can be used for sorting or collapsing matching documents.

Parameters

- **doc** Document object to build a key for.

Implemented in Xapian::LatLongDistanceKeyMaker, and Xapian::MultiValueKeyMaker.

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

- xapian/keymaker.h

7.34 Xapian::LatLongCoord Struct Reference

A latitude-longitude coordinate.

Public Member Functions

- **LatLongCoord ()**
  Construct an uninitialised coordinate.
- **LatLongCoord (double latitude, double longitude)**
  Construct a coordinate.
- **void unserialise (const std::string &serialised)**
  Unserialise a string and set this object to its coordinate.
- **void unserialise (const char *ptr, const char *end)**
  Unserialise a buffer and set this object to its coordinate.
- **std::string serialise () const**
  Return a serialised representation of the coordinate.
- **bool operator< (const LatLongCoord &other) const**
  Compare with another LatLongCoord.
- **std::string get_description () const**
  Return a string describing this object.
Public Attributes

- double latitude
  A latitude, as decimal degrees.
- double longitude
  A longitude, as decimal degrees.

7.34.1 Detailed Description

A latitude-longitude coordinate.

Experimental - see http://xapian.org/docs/deprecation#experimental-features

Note that latitude-longitude coordinates are only precisely meaningful if the datum used to define them is specified. This class ignores this issue - it is up to the caller to ensure that the datum used for each coordinate in a system is consistent.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 Xapian::LatLongCoord::LatLongCoord ( double latitude, double longitude )

Construct a coordinate.

If the supplied longitude is out of the standard range, it will be normalised to the range 0 <= longitude < 360.

If you want to avoid the checks (for example, you know that your values are already in range), you can use the alternate constructor to construct an uninitialised coordinate, and then set the latitude and longitude directly.

Exceptions

InvalidArgumentError the supplied latitude is out of range.

7.34.3 Member Function Documentation

7.34.3.1 void Xapian::LatLongCoord::unserialise ( const std::string & serialised )

Unserialise a string and set this object to its coordinate.

Parameters

serialised the string to unserialise the coordinate from.

Exceptions

Xapian::SerialisationError if the string does not contain a valid serialised latitude-longitude pair, or contains extra data at the end of it.
7.35 Xapian::LatLongCoords Class Reference

A sequence of latitude-longitude coordinates.

7.34.3.2 void Xapian::LatLongCoord::unserialise ( const char ** ptr, const char * end )

Unserialise a buffer and set this object to its coordinate.
The buffer may contain further data after that for the coordinate.

Parameters

<table>
<thead>
<tr>
<th>ptr</th>
<th>A pointer to the start of the string. This will be updated to point to the end of the data representing the coordinate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>end</td>
<td>A pointer to the end of the string.</td>
</tr>
</tbody>
</table>

Exceptions

Xapian::SerialisationError if the string does not start with a valid serialised latitude-longitude pair.

7.34.4 Member Data Documentation

7.34.4.1 double Xapian::LatLongCoord::latitude

A latitude, as decimal degrees.
Should be in the range -90 <= latitude <= 90
Positive latitudes represent the northern hemisphere.
Referenced by operator<().

7.34.4.2 double Xapian::LatLongCoord::longitude

A longitude, as decimal degrees.
Will be wrapped around, so for example, -150 is equal to 210. When obtained from a serialised form, will be in the range 0 <= longitude < 360.
Longitudes increase as coordinates move eastwards.
Referenced by operator<().

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

- xapian/geospatial.h

7.35 Xapian::LatLongCoords Class Reference

A sequence of latitude-longitude coordinates.
Public Member Functions

- **LatLongCoordsIterator begin () const**
  Get a begin iterator for the coordinates.

- **LatLongCoordsIterator end () const**
  Get an end iterator for the coordinates.

- **size_t size () const**
  Get the number of coordinates in the container.

- **bool empty () const**
  Return true if and only if there are no coordinates in the container.

- **void append (const LatLongCoord &coord)**
  Append a coordinate to the end of the sequence.

- **LatLongCoords ()**
  Construct an empty container.

- **LatLongCoords (const LatLongCoord &coord)**
  Construct a container holding one coordinate.

- **void unserialise (const std::string &serialised)**
  Unserialise a string and set this object to the coordinates in it.

- **std::string serialise () const**
  Return a serialised form of the coordinate list.

- **std::string get_description () const**
  Return a string describing this object.

### 7.35.1 Detailed Description

A sequence of latitude-longitude coordinates.

Experimental - see [http://xapian.org/docs/deprecation#experimental-features](http://xapian.org/docs/deprecation#experimental-features)

### 7.35.2 Member Function Documentation

#### 7.35.2.1 void Xapian::LatLongCoords::unserialise ( const std::string & serialised )

Unserialise a string and set this object to the coordinates in it.

**Parameters**

| serialised | the string to unserialise the coordinates from. |

**Exceptions**

| Xapian::Serialisation::Error | if the string does not contain a valid serialised latitude-longitude pair, or contains junk at the end of it. |

The documentation for this class was generated from the following file:
7.36 Xapian::LatLongCoordsIterator Class Reference

An iterator across the values in a LatLongCoords object.

Public Member Functions

- **LatLongCoordsIterator ()**
  
  Default constructor. Produces an uninitialised iterator.

7.36.1 Detailed Description

An iterator across the values in a LatLongCoords object.

Experimental - see [http://xapian.org/docs/deprecation#experimental-features](http://xapian.org/docs/deprecation#experimental-features)

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

- xapian/geospatial.h

7.37 Xapian::LatLongDistanceKeyMaker Class Reference

KeyMaker subclass which sorts by distance from a latitude/longitude.

Inheritance diagram for Xapian::LatLongDistanceKeyMaker:

```
Xapian::KeyMaker

Xapian::LatLongDistanceKeyMaker
```

Public Member Functions

- std::string **operator()** (const Xapian::Document &doc) const
Build a key string for a Document.

### 7.37.1 Detailed Description

**KeyMaker** subclass which sorts by distance from a latitude/longitude.

**Experimental** - see [http://xapian.org/docs/deprecation#experimental-features](http://xapian.org/docs/deprecation#experimental-features)

Results are ordered by the distance from a fixed point, or list of points, calculated according to the metric supplied. If multiple points are supplied (either in the constructor, or in the coordinates stored in a document), the closest pointwise distance is used.

If a document contains no coordinate stored in the specified slot, a special value for the distance will be used. This defaults to a large number, so that such results get a low rank, but may be specified by a constructor parameter.

### 7.37.2 Member Function Documentation

#### 7.37.2.1 std::string Xapian::LatLongDistanceKeyMaker::operator() ( const Xapian::Document & doc ) const [virtual]

Build a key string for a Document.

These keys can be used for sorting or collapsing matching documents.

**Parameters**

| doc | Document object to build a key for. |

Implements **Xapian::KeyMaker**.

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

- xapian/geospatial.h

---

### 7.38 Xapian::LatLongDistancePostingSource Class Reference

Posting source which returns a weight based on geospatial distance.
Inheritance diagram for Xapian::LatLongDistancePostingSource:

```
Xapian::PostingSource
  ↓
Xapian::ValuePostingSource
  ↓
Xapian::LatLongDistancePostingSource
```

Public Member Functions

- **LatLongDistancePostingSource (Xapian::valueno slot_, const LatLongCoords &centre_, const LatLongMetric &metric_, double max_range_=0.0, double k1_=1000.0, double k2_=1.0)**
  
  Construct a new match decider which returns only documents within range of one of the central coordinates.

- **void next (double min_wt)**
  
  Advance the current position to the next matching document.

- **void skip_to (Xapian::docid min_docid, double min_wt)**
  
  Advance to the specified docid.

- **bool check (Xapian::docid min_docid, double min_wt)**
  
  Check if the specified docid occurs.

- **double get_weight () const**
  
  Return the weight contribution for the current document.

- **LatLongDistancePostingSource * clone () const**
  
  Clone the posting source.

- **std::string name () const**
  
  Name of the posting source class.

- **std::string serialise () const**
  
  Serialise object parameters into a string.

- **LatLongDistancePostingSource * unserialise_with_registry (const std::string &, const Registry &registry) const**
  
  Create object given string serialisation returned by serialise().
• void \texttt{init (const Database &db_)}
  \begin{quote}
  Set this \texttt{PostingSource} to the start of the list of postings.
  \end{quote}
• std::string \texttt{get\_description () const}
  \begin{quote}
  Return a string describing this object.
  \end{quote}

7.38.1 Detailed Description

Posting source which returns a weight based on geospatial distance.

Experimental - see http://xapian.org/docs/deprecation#experimental-features

Results are weighted by the distance from a fixed point, or list of points, calculated
according to the metric supplied. If multiple points are supplied (either in the constructor,
or in the coordinates stored in a document), the closest pointwise distance is used.

Documents further away than a specified maximum range (or with no location stored in
the specified slot) will not be returned.

The weight returned is computed from the distance using the formula:

\[ k_1 \times \text{pow}(\text{distance} + k_1, -k_2) \]

(Where \( k_1 \) and \( k_2 \) are (strictly) positive, floating point constants, which default to 1000
and 1, respectively. Distance is measured in metres, so this means that something at
the centre gets a weight of 1.0, something 1km away gets a weight of 0.5, and something
3km away gets a weight of 0.25, etc)

7.38.2 Constructor & Destructor Documentation

7.38.2.1

\begin{verbatim}
Xapian::LatLongDistancePostingSource::LatLongDistancePostingSource (Xapian::valueno slot_,
  const LatLongCoords &centre_, const LatLongMetric &metric_,
  double max_range_=0.0, double k1_=1000.0, double k2_=1.0 )
\end{verbatim}

Construct a new match decider which returns only documents within range of one of the
central coordinates.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{db_}</td>
<td>The database to read values from.</td>
</tr>
<tr>
<td>\texttt{slot_}</td>
<td>The value slot to read values from.</td>
</tr>
<tr>
<td>\texttt{centre_}</td>
<td>The centre point to use for distance calculations.</td>
</tr>
<tr>
<td>\texttt{metric_}</td>
<td>The metric to use for distance calculations.</td>
</tr>
<tr>
<td>\texttt{max_range_}</td>
<td>The maximum distance for documents which are returned.</td>
</tr>
<tr>
<td>\texttt{k1_}</td>
<td>The \texttt{k1} constant to use in the weighting function.</td>
</tr>
<tr>
<td>\texttt{k2_}</td>
<td>The \texttt{k2} constant to use in the weighting function.</td>
</tr>
</tbody>
</table>

7.38.3 Member Function Documentation

---

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.38.3.1 bool Xapian::LatLongDistancePostingSource::check ( Xapian::docid did, double min_wt ) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id did actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:

- return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after did.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost.

Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn’t.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the init() method for details.

Parameters

<table>
<thead>
<tr>
<th>did</th>
<th>The document id to check.</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_wt</td>
<td>The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).</td>
</tr>
</tbody>
</table>

Reimplemented from Xapian::ValuePostingSource.

7.38.3.2 LatLongDistancePostingSource* Xapian::LatLongDistancePostingSource::clone ( ) const [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.
Note that the returned object will be deallocated by Xapian after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

Reimplemented from Xapian::PostingSource.

7.38.3.3 std::string Xapian::LatLongDistancePostingSource::get_description ( ) const [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.

7.38.3.4 double Xapian::LatLongDistancePostingSource::get_weight ( ) const [virtual]

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.

7.38.3.5 void Xapian::LatLongDistancePostingSource::init ( const Database & db ) [virtual]

Set this PostingSource to the start of the list of postings.

This is called automatically by the matcher prior to each query being processed.

If a PostingSource is used for multiple searches, init() will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters

| db | The database which the PostingSource should iterate through. |

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.
Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using clone()), and each PostingSource will be passed one of the sub-databases as the db parameter here. The db parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValuePostingSource.

### 7.38.3.6 std::string Xapian::LatLongDistancePostingSource::name ( ) const

[virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "-FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::-FooPostingSource".

This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.

Reimplemented from Xapian::PostingSource.

### 7.38.3.7 void Xapian::LatLongDistancePostingSource::next ( double min_wt )

[virtual]

Advance the current position to the next matching document.

The PostingSource starts before the first entry in the list, so next() must be called before any methods which need the context of the current position.

Xapian will always call init() on a PostingSource before calling this for the first time.

**Parameters**

- **min_wt**  
  The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented from Xapian::ValuePostingSource.

### 7.38.3.8 std::string Xapian::LatLongDistancePostingSource::serialise ( ) const

[virtual]

Serialise object parameters into a string.
The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::PostingSource.

7.38.3.9 void Xapian::LatLongDistancePostingSource::skip_to ( Xapian::docid did, double min_wt ) [virtual]

Advance to the specified docid.

If the specified docid isn’t in the list, position ourselves on the first document after it (or at_end()) if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the init() method for details.

Parameters

<table>
<thead>
<tr>
<th>did</th>
<th>The document id to advance to.</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_wt</td>
<td>The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).</td>
</tr>
</tbody>
</table>

Reimplemented from Xapian::ValuePostingSource.

7.38.3.10 LatLongDistancePostingSource* Xapian::LatLongDistancePostingSource::unserialise_with_registry ( const std::string & s, const Registry & registry ) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.-org/ticket/554#comment:1

This method is supplied with a Registry object, which can be used when unserialising objects contained within the posting source. The default implementation simply calls unserialise() which doesn’t take the Registry object, so you do not need to implement this
method unless you want to take advantage of the Registry object when unserialising.

Parameters

- `s` A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

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

- xapian/geospatial.h

### 7.39 Xapian::LatLongMetric Class Reference

Base class for calculating distances between two lat/long coordinates.

Inheritance diagram for Xapian::LatLongMetric:

```
Xapian::LatLongMetric
  ↘
  Xapian::GreatCircleMetric
```

#### Public Member Functions

- Virtual `~LatLongMetric()` 
  Destructor.
- Virtual double `pointwise_distance(const LatLongCoord &a, const LatLongCoord &b) const` 
  Return the distance between two coordinates, in metres.
- `operator()(const LatLongCoords &a, const LatLongCoords &b) const` 
  Return the distance between two coordinate lists, in metres.
- `operator()(const LatLongCoords &a, const std::string &b) const` 
  Return the distance between two coordinate lists, in metres.
- `operator()(const LatLongCoords &a, const char *b_ptr, size_t b_len) const` 
  Return the distance between two coordinate lists, in metres.
Return the distance between two coordinate lists, in metres.

- virtual LatLongMetric * clone() const = 0
  
  Clone the metric.

- virtual std::string name() const = 0
  
  Return the full name of the metric.

- virtual std::string serialise() const = 0
  
  Serialise object parameters into a string.

- virtual LatLongMetric * unserialise(const std::string &s) const = 0
  
  Create object given string serialisation returned by serialise().

### 7.39.1 Detailed Description

Base class for calculating distances between two lat/long coordinates.

Experimental - see [http://xapian.org/docs/deprecation#experimental-features](http://xapian.org/docs/deprecation#experimental-features)

### 7.39.2 Member Function Documentation

#### 7.39.2.1 virtual LatLongMetric * Xapian::LatLongMetric::clone() const [pure virtual]

Clone the metric.

Implemented in Xapian::GreatCircleMetric.

#### 7.39.2.2 virtual std::string Xapian::LatLongMetric::name() const [pure virtual]

Return the full name of the metric.

This is used when serialising and unserialising metrics; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a LatLongMetric subclass called "FooLatLongMetric" in the "Xapian" namespace the result of this call should be "Xapian::FooLatLongMetric".

Implemented in Xapian::GreatCircleMetric.

#### 7.39.2.3 double Xapian::LatLongMetric::operator()( const LatLongCoords & a, const LatLongCoords & b ) const

Return the distance between two coordinate lists, in metres.

The distance between the coordinate lists is defined to be the minimum pairwise distance between coordinates in the lists.
## 7.39 Xapian::LatLongMetric Class Reference

### Exceptions

*InvalidArgumentError* either of the lists is empty.

### Parameters

| a | The first coordinate list. |
| b | The second coordinate list. |

## 7.39.2.4 double Xapian::LatLongMetric::operator() ( const LatLongCoords & a, const std::string & b ) const [inline]

Return the distance between two coordinate lists, in metres.

One of the coordinate lists is supplied in serialised form.

The distance between the coordinate lists is defined to be the minimum pairwise distance between coordinates in the lists.

### Exceptions

*InvalidArgumentError* either of the lists is empty.

### Parameters

| a | The first coordinate list. |
| b | The second coordinate list, in serialised form. |

References Xapian::LatLongCoords::size().

## 7.39.2.5 double Xapian::LatLongMetric::operator() ( const LatLongCoords & a, const char * b_ptr, size_t b_len ) const

Return the distance between two coordinate lists, in metres.

One of the coordinate lists is supplied in serialised form.

The distance between the coordinate lists is defined to be the minimum pairwise distance between coordinates in the lists.

### Exceptions

*InvalidArgumentError* either of the lists is empty.

### Parameters

| a | The first coordinate list. |
| b_ptr | The start of the serialised form of the second coordinate list. |
| b_len | The length of the serialised form of the second coordinate list. |
7.39.2.6 virtual std::string Xapian::LatLongMetric::serialise() const [pure virtual]

Serialise object parameters into a string.
The serialised parameters should represent the configuration of the metric.
Implemented in Xapian::GreatCircleMetric.

7.39.2.7 virtual LatLongMetric* Xapian::LatLongMetric::unserialise(const std::string &s) const [pure virtual]

Create object given string serialisation returned by serialise().

Parameters

| s | A serialised instance of this LatLongMetric subclass. |

Implemented in Xapian::GreatCircleMetric.
The documentation for this class was generated from the following file:

- xapian/geospatial.h

7.40 Xapian::LogicError Class Reference

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

Inheritance diagram for Xapian::LogicError:

```
Xapian::LogicError
  Xapian::AssertionError
  Xapian::InvalidArgumentError
  Xapian::InvalidOperationError
  Xapian::UnimplementedError
  Xapian::Error
```

7.40.1 Detailed Description

The base class for exceptions indicating errors in the program logic.
A subclass of `LogicError` will be thrown if Xapian detects a violation of a class invariant or a logical precondition or postcondition, etc.

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

- `xapian/error.h`

### 7.41 Xapian::MatchDecider Class Reference

Base class for matcher decision functor.

Inheritance diagram for Xapian::MatchDecider:

```
Xapian::MatchDecider
Xapian::ValueSetMatchDecider
```

#### 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.

### 7.41.1 Detailed Description

Base class for matcher decision functor.

### 7.41.2 Member Function Documentation

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

Decide whether we want this document to be in the MSet.
Class Documentation

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc</td>
<td>The document to test.</td>
</tr>
</tbody>
</table>

Returns

true if the document is acceptable, or false if the document should be excluded from the MSet.

Implemented in Xapian::ValueSetMatchDecider.

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

- xapian/enquire.h

7.42 Xapian::MatchSpy Class Reference

Abstract base class for match spies.

Inheritance diagram for Xapian::MatchSpy:

```
Xapian::MatchSpy
    \--- Xapian::ValueCountMatchSpy
```

Public Member Functions

- virtual ~MatchSpy ()
  
  Virtual destructor, because we have virtual methods.

- virtual void operator() (const Xapian::Document &doc, double wt)=0
  
  Register a document with the match spy.

- virtual MatchSpy * clone () const
  
  Clone the match spy.

- virtual std::string name () const
  
  Return the name of this match spy.

- virtual std::string serialise () const
Return this object’s parameters serialised as a single string.

- virtual **MatchSpy** * unserialise (const std::string &s, const Registry &context) const
  
  Unserialise parameters.
- virtual std::string serialise_results () const
  
  Serialise the results of this match spy.
- virtual void merge_results (const std::string &s)
  
  Unserialise some results, and merge them into this matchspy.
- virtual std::string get_description () const
  
  Return a string describing this object.

Protected Member Functions

- **MatchSpy** ()
  
  Default constructor, needed by subclass constructors.

7.42.1 Detailed Description

Abstract base class for match spies.

The subclasses will generally accumulate information seen during the match, to calculate aggregate functions, or other profiles of the matching documents.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 virtual Xapian::MatchSpy::~MatchSpy ( ) [virtual]

Virtual destructor, because we have virtual methods.

7.42.3 Member Function Documentation

7.42.3.1 virtual **MatchSpy** & Xapian::MatchSpy::clone ( ) const [virtual]

Clone the match spy.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be passed information about the results seen by the parent.

If you don’t want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: [http://trac.xapian.org/ticket/554#comment:1](http://trac.xapian.org/ticket/554#comment:1)

Reimplemented in Xapian::ValueCountMatchSpy.
7.42.3.2 virtual std::string Xapian::MatchSpy::get_description() const [virtual]

Return a string describing this object.
This default implementation returns a generic answer, to avoid forcing those deriving
their own MatchSpy subclasses from having to implement this (they may not care what
get_description() gives for their subclass).
Reimplemented in Xapian::ValueCountMatchSpy.

7.42.3.3 virtual void Xapian::MatchSpy::merge_results(const std::string &s) [virtual]

Unserialize some results, and merge them into this matchspy.
The order in which results are merged should not be significant, since this order is not
specified (and will vary depending on the speed of the search in each sub-database).
If you don’t want to support the remote backend in your match spy, you can use the
default implementation which simply throws Xapian::UnimplementedError.

Parameters

\[ s \quad \text{A string containing the serialised results.} \]

Reimplemented in Xapian::ValueCountMatchSpy.

7.42.3.4 virtual std::string Xapian::MatchSpy::name() const [virtual]

Return the name of this match spy.
This name is used by the remote backend. It is passed with the serialised parameters
to the remote server so that it knows which class to create.
Return the full namespace-qualified name of your class here - if your class is called
MyApp::FooMatchSpy, return "MyApp::FooMatchSpy" from this method.
If you don’t want to support the remote backend in your match spy, you can use the
default implementation which simply throws Xapian::UnimplementedError.
Reimplemented in Xapian::ValueCountMatchSpy.

7.42.3.5 virtual void Xapian::MatchSpy::operator()(const Xapian::Document &doc, double wt) [pure virtual]

Register a document with the match spy.
This is called by the matcher once with each document seen by the matcher during the
match process. Note that the matcher will often not see all the documents which match
the query, due to optimisations which allow low-weighted documents to be skipped, and
allow the match process to be terminated early.
Parameters

<table>
<thead>
<tr>
<th><code>doc</code></th>
<th>The document seen by the match spy.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wt</code></td>
<td>The weight of the document.</td>
</tr>
</tbody>
</table>

Implemented in `Xapian::ValueCountMatchSpy`.

7.42.3.6 virtual `std::string` `Xapian::MatchSpy::serialise ( ) const` [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Reimplemented in `Xapian::ValueCountMatchSpy`.

7.42.3.7 virtual `std::string` `Xapian::MatchSpy::serialise_results ( ) const` [virtual]

Serialise the results of this match spy.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Reimplemented in `Xapian::ValueCountMatchSpy`.

7.42.3.8 virtual `MatchSpy*` `Xapian::MatchSpy::unserialise ( const `std::string`& `s`, `const` `Registry`& `context` ) const` [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the `serialise()` method and allocates and returns a new object initialised with them.

If you don't want to support the remote backend in your match spy, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Note that the returned object will be deallocated by `Xapian` after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the `Xapian` API for use from another language) then you can define a static `operator delete` method in your subclass as shown here: [http://trac.xapian.org/ticket/554#comment:1](http://trac.xapian.org/ticket/554#comment:1)

Parameters

<table>
<thead>
<tr>
<th><code>s</code></th>
<th>A string containing the serialised results.</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>context</code></td>
<td><code>Registry</code> object to use for unserialisation to permit <code>MatchSpy</code> sub-classes with sub-MatchSpy objects to be implemented.</td>
</tr>
</tbody>
</table>

Reimplemented in `Xapian::ValueCountMatchSpy`.

The documentation for this class was generated from the following file:
• xapian/matchspy.h

7.43 Xapian::MSet Class Reference

A match set (MSet).

Public Types

• typedef MSetIterator value_type
  Allow use as an STL container.
• typedef MSetIterator iterator
  Allow use as an STL container.
• typedef MSetIterator const_iterator
  Allow use as an STL container.
• typedef MSetIterator & reference
  Allow use as an STL container.
• typedef MSetIterator & const_reference
  Allow use as an STL container.
• typedef MSetIterator ∗ pointer
  Allow use as an STL container.
• typedef Xapian::doccount_diff difference_type
  Allow use as an STL container.
• typedef Xapian::doccount size_type
  Allow use as an STL container.

Public Member Functions

• MSet ()
  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.
• int convert_to_percent (double wt) const
This converts the weight supplied to a percentage score.

- **int 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.

- **double 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::doccount get_uncollapsed_matches_lower_bound () const**
  A lower bound on the number of documents in the database which would match the query if collapsing wasn’t used.

- **Xapian::doccount get_uncollapsed_matches_estimated () const**
  A estimate of the number of documents in the database which would match the query if collapsing wasn’t used.

- **Xapian::doccount get_uncollapsed_matches_upper_bound () const**
  A upper bound on the number of documents in the database which would match the query if collapsing wasn’t used.

- **double get_max_possible () const**
  The maximum possible weight in the MSet.

- **double 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 items 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.
7.43.1 Detailed Description

A match set (MSet).
This class represents (a portion of) the results of a query.

7.43.2 Member Function Documentation

7.43.2.1 int Xapian::MSet::convert_to_percent ( double 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.

Parameters

| wt | The weight to convert. |

7.43.2.2 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. |

7.43.2.3 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.

7.43.2.4 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.

7.43.2.5 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.

7.43.2.6 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.

7.43.2.7 double Xapian::MSet::get_max_attained ( ) const
The greatest weight which is attained by any document in the database. If firstitem == 0 and the primary ordering is by relevance, this is the weight of the first entry in the MSet. If no documents are found by the query, this will be 0. Note that calculation of max_attained requires calculation of at least one result item - therefore, if no items were requested when the query was performed (by specifying maxitems = 0 in Xapian::Enquire::get_mset()), this value will be 0.

7.43.2.8 double 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.

7.43.2.9 Xapian::doccount Xapian::MSet::get_termfreq ( const std::string & tname ) const
Return the term frequency of the given query term.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tname</code></td>
<td>The term to look for.</td>
</tr>
</tbody>
</table>

This is sometimes more efficient than asking the database directly for the term frequency - in particular, if the term was in the query, its frequency will usually be cached in the MSet.

#### 7.43.2.10 `double Xapian::MSet::get_termweight ( const std::string & tname ) const`

Return the term weight of the given query term.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tname</code></td>
<td>The term to look for.</td>
</tr>
</tbody>
</table>

**Exceptions**

- *Xapian::InvalidArgumentError* is thrown if the term was not in the query.

#### 7.43.2.11 `Xapian::doccount Xapian::MSet::max_size ( ) const [inline]

Required to allow use as an STL container.

#### 7.43.2.12 `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.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>i</code></td>
<td>The index into the MSet.</td>
</tr>
</tbody>
</table>

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

- xapian/enquire.h

### 7.44 Xapian::MSetIterator Class Reference

An iterator pointing to items in an MSet.
Public Types

- typedef std::bidirectional_iterator_tag iterator_category
  
  Allow use as an STL iterator.

- typedef Xapian::docid value_type

  Allow use as an STL iterator.

- typedef Xapian::doccount_diff difference_type

  Allow use as an STL iterator.

- typedef Xapian::docid * pointer

  Allow use as an STL iterator.

- typedef Xapian::docid & reference

  Allow use as an STL iterator.

Public Member Functions

- MSetIterator ()

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

- MSetIterator (const MSetIterator & other)

  Copying is allowed (and is cheap).

- void operator= (const MSetIterator & other)

  Assignment is allowed (and is cheap).

- MSetIterator & operator++ ()

  Advance the iterator.

- MSetIterator operator++ (int)

  Advance the iterator (postfix variant).

- MSetIterator & operator-- ()

  Decrement the iterator.

- MSetIterator operator-- (int)

  Decrement the iterator (postfix variant).

- Xapian::docid operator* () const

  Get the document ID for the current position.

- Xapian::Document get_document () const

  Get a Xapian::Document object for the current position.

- Xapian::doccount get_rank () const

  Get the rank of the document at the current position.

- double 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.

- int get_percent () const

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

- std::string get_description () const

  Return a string describing this object.
Friends

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

7.44.1 Detailed Description

An iterator pointing to items in an MSet.
This is used for access to individual results of a match.

7.44.2 Member Function Documentation

7.44.2.1 Xapian::doccount Xapian::MSetIterator::getCollapseCount() 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 col-
lapse 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).

7.44.2.2 Xapian::Document Xapian::MSetIterator::getDocument() 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 en-
sure 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. |
7.44.2.3  int Xapian::MSetIterator::get_percent ( ) const

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

You probably don’t want to show these percentage scores to end users in new applications - they’re not really a percentage of anything meaningful, and research seems to suggest that users don’t find numeric scores in search results useful.

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

The intention is that the highest weighted document will get 100 if it matches all the weight-contributing terms in the query. However, currently it may get a lower percentage score if you use a MatchDecider and the sorting is primarily by value. In this case, the percentage for a particular document may vary depending on the first, max_size, and checkatleast parameters passed to Enquire::get_mset() (this bug is hard to fix without having to apply the MatchDecider to potentially many more documents, which is potentially costly).

7.44.2.4  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.

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

- xapian/enquire.h

7.45  Xapian::MultiValueKeyMaker Class Reference

KeyMaker subclass which combines several values.

Inheritance diagram for Xapian::MultiValueKeyMaker:
Public Member Functions

- virtual std::string operator() (const Xapian::Document &doc) const
  Build a key string for a Document.

7.45.1 Detailed Description

KeyMaker subclass which combines several values.

When the result is used for sorting, 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. If reverse is true for a value, then the sort order for that value is reversed.

When used for collapsing, the documents will only be considered equal if all the values specified match. If none of the specified values are set then the generated key will be empty, so such documents won’t be collapsed (which is consistent with the behaviour in the "collapse on a value" case). If you’d prefer that documents with none of the keys set are collapsed together, then you can set reverse for at least one of the values. Other than this, it isn’t useful to set reverse for collapsing.

7.45.2 Member Function Documentation

7.45.2.1 virtual std::string Xapian::MultiValueKeyMaker::operator() ( const Xapian::Document & doc ) const [virtual]

Build a key string for a Document.

These keys can be used for sorting or collapsing matching documents.

Parameters

| doc | Document object to build a key for. |

Implements Xapian::KeyMaker.

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

- xapian/keymaker.h

7.46 Xapian::NetworkError Class Reference

Indicates a problem communicating with a remote database.
Inheritance diagram for Xapian::NetworkError:

```
Xapian::Error
   `--> Xapian::RuntimeError
       `--> Xapian::NetworkError
               `--> Xapian::NetworkTimeoutError
```

Public Member Functions

- NetworkError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.

- NetworkError (const std::string &msg_, int errno_)
  
  Construct from message and errno value.

7.46.1 Detailed Description

Indicates a problem communicating with a remote database.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 Xapian::NetworkError::NetworkError ( const std::string & msg_, const std::string & context_=std::string(), int errno_=0 ) [inline, explicit]

General purpose constructor.
Parameters

- `msg_` Message giving details of the error, intended for human consumption.
- `context_` Optional context information for this error.
- `errno_` Optional errno value associated with this error.

### 7.46.2.2 Xapian::NetworkError

**Xapian::NetworkError::NetworkError** (const std::string &`msg_`, int `errno_`)

[inline]

Constructor from message and errno value.

Parameters

- `msg_` Message giving details of the error, intended for human consumption.
- `errno_` Optional errno value associated with this error.

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

- xapian/error.h

### 7.47 Xapian::NetworkTimeoutError

Indicates a timeout expired while communicating with a remote database.
Inheritance diagram for Xapian::NetworkTimeoutError:

```
Xapian::Error
    Xapian::RuntimeError
        Xapian::NetworkError
            Xapian::NetworkTimeoutError
```

Public Member Functions

- `NetworkTimeoutError` (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  General purpose constructor.
- `NetworkTimeoutError` (const std::string &msg_, int errno_)
  
  Construct from message and errno value.

### 7.47.1 Detailed Description

Indicates a timeout expired while communicating with a remote database.

### 7.47.2 Constructor & Destructor Documentation

#### 7.47.2.1 Xapian::NetworkTimeoutError::NetworkTimeoutError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0) [inline, explicit]

General purpose constructor.
Parameters

| Message giving details of the error, intended for human consumption. |
| Optional context information for this error. |
| Optional errno value associated with this error. |

7.47.2.2 Xapian::NetworkTimeoutError::NetworkTimeoutError ( const std::string & msg_, int errno_ ) [inline]
Construct from message and errno value.

Parameters

| Message giving details of the error, intended for human consumption. |
| Optional errno value associated with this error. |

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

- xapian/error.h

7.48 Xapian::NumberValueRangeProcessor Class Reference

Handle a number range.

Inheritance diagram for Xapian::NumberValueRangeProcessor:
Public Member Functions

- **NumberValueRangeProcessor** (Xapian::valueno slot_)
  
  Constructor.

- **NumberValueRangeProcessor** (Xapian::valueno slot_, const std::string &str_, bool prefix_=true)
  
  Constructor.

- **Xapian::valueno operator()** (std::string &begin, std::string &end)

  Check for a valid numeric range.

### 7.48.1 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).

### 7.48.2 Constructor & Destructor Documentation

#### 7.48.2.1 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno slot_) [inline]

Constructor.

Parameters

| slot_ | The value number to return from operator(). |

#### 7.48.2.2 Xapian::NumberValueRangeProcessor::NumberValueRangeProcessor (Xapian::valueno slot_, const std::string &str_, bool prefix_=true) [inline]

Constructor.

Parameters

| slot_ | 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 num-
bers. (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.

7.48.3 Member Function Documentation

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

Check for a valid numeric range.

Parameters

| in, out | begin  | The start of the range as specified in the query string by the user. This parameter is a non-const reference so the Value-
|         |        | RangeProcessor can modify it to return the value to start the range with. |
| in, out | end    | The end of the range. This is also a non-const reference so it can be modified. |

Returns

If BEGIN..END is a valid numeric range with the specified prefix/suffix (if one was
specified), this method modifies them by removing the prefix/suffix, converting to
a number, and encoding with Xapian::sortable_serialise(), and returns the value of
slot_ passed at construction time. Otherwise it returns Xapian::BAD_VALUENO.

Reimplemented from Xapian::StringValueRangeProcessor.

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

- xapian/queryparser.h

7.49 Xapian::PositionIterator Class Reference

Class for iterating over term positions.

Public Member Functions

- PositionIterator (const PositionIterator &o)
Copy constructor.

- `PositionIterator & operator= (const PositionIterator &o)`
  Assignment.

- `PositionIterator ()`
  Default constructor.

- `~PositionIterator ()`
  Destructor.

- `Xapian::termpos operator* () const`
  Return the term position at the current iterator position.

- `PositionIterator & operator++ ()`
  Advance the iterator to the next position.

- `DerefWrapper_<Xapian::termpos, operator++ (int)>
  Advance the iterator to the next position (postfix version).

- `void skip_to (Xapian::termpos termpos)`
  Advance the iterator to term position `termpos`.

- `std::string get_description () const`
  Return a string describing this object.

### 7.49.1 Detailed Description

Class for iterating over term positions.

### 7.49.2 Constructor & Destructor Documentation

#### 7.49.2.1 Xapian::PositionIterator::PositionIterator ( ) [inline]

Default constructor.

Creates an uninitialised iterator, which can’t be used before being assigned to, but is sometimes syntactically convenient.

### 7.49.3 Member Function Documentation

#### 7.49.3.1 void Xapian::PositionIterator::skip_to ( Xapian::termpos termpos )

Advance the iterator to term position `termpos`.

**Parameters**

| `termpos` | The position to advance to. If this position isn’t in the stream being iterated, then the iterator is moved to the next term position after it which is. |

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

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
Class for iterating over a list of terms.

Public Member Functions

- **PostingIterator** (const PostingIterator &o)
  
  Copy constructor.

- **PostingIterator & operator=** (const PostingIterator &o)
  
  Assignment.

- **PostingIterator ()**
  
  Default constructor.

- **~PostingIterator ()**
  
  Destructor.

- **Xapian::docid operator* () const**
  
  Return the document id at the current position.

- **Xapian::termcount get_wdf () const**
  
  Return the wdf for the document at the current position.

- **Xapian::termcount get_doclength () const**
  
  Return the length of the document at the current position.

- **PositionIterator positionlist_begin () const**
  
  Return a PositionIterator for the current document.

- **PositionIterator positionlist_end () const**
  
  Return an end PositionIterator for the current document.

- **PostingIterator & operator++ ()**
  
  Advance the iterator to the next position.

- **DerefWrapper_< Xapian::docid > operator++ (int)**
  
  Advance the iterator to the next position (postfix version).

- **void skip_to (Xapian::docid did)**
  
  Advance the iterator to document did.

- **std::string get_description () const**
  
  Return a string describing this object.

7.50.1 Detailed Description

Class for iterating over a list of terms.
7.50.2 Constructor & Destructor Documentation

7.50.2.1 Xapian::PostingIterator::PostingIterator ( ) [inline]

Default constructor.
Creates an uninitialised iterator, which can't be used before being assigned to, but is sometimes syntactically convenient.

7.50.3 Member Function Documentation

7.50.3.1 void Xapian::PostingIterator::skip_to ( Xapian::docid did )

Advance the iterator to document did.

Parameters

- **did**: The document id to advance to. If this document id isn't in the stream being iterated, then the iterator is moved to the next document id after it which is.

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

- xapian/postingiterator.h

7.51 Xapian::PostingSource Class Reference

Base class which provides an "external" source of postings.

Inheritance diagram for Xapian::PostingSource:

```
Xapian::PostingSource
\-- Xapian::FixedWeightPostingSource
\-- Xapian::ValuePostingSource
\-- Xapian::LatLongDistancePostingSource
\-- Xapian::ValueMapPostingSource
\-- Xapian::ValueWeightPostingSource
\-- Xapian::DecreasingValueWeightPostingSource
```

Public Member Functions

- virtual Xapian::doccount get_termfreq_min () const =0
  
  *A lower bound on the number of documents this object can return.*

- virtual Xapian::doccount get_termfreq_est () const =0
  
  *An estimate of the number of documents this object can return.*

- virtual Xapian::doccount get_termfreq_max () const =0
  
  *An upper bound on the number of documents this object can return.*
• double get_maxweight () const
  
  Return the currently set upper bound on what get_weight() can return.

• virtual double get_weight () const
  
  Return the weight contribution for the current document.

• virtual Xapian::docid get_docid () const =0
  
  Return the current docid.

• virtual void next (double min_wt)=0
  
  Advance the current position to the next matching document.

• virtual void skip_to (Xapian::docid did, double min_wt)
  
  Advance to the specified docid.

• virtual bool check (Xapian::docid did, double min_wt)
  
  Check if the specified docid occurs.

• virtual bool at_end () const =0
  
  Return true if the current position is past the last entry in this list.

• virtual PostingSource + clone () const
  
  Clone the posting source.

• virtual std::string name () const
  
  Name of the posting source class.

• virtual std::string serialise () const
  
  Serialise object parameters into a string.

• virtual PostingSource + unserialise (const std::string &s) const
  
  Create object given string serialisation returned by serialise().

• virtual PostingSource + unserialise_with_registry (const std::string &s, const -
  Registry &registry) const
  
  Create object given string serialisation returned by serialise().

• virtual void init (const Database &db)=0
  
  Set this PostingSource to the start of the list of postings.

• virtual std::string get_description () const
  
  Return a string describing this object.

Protected Member Functions

• PostingSource ()
  
  Allow subclasses to be instantiated.

• void set_maxweight (double max_weight)
  
  Set an upper bound on what get_weight() can return from now on.

7.51.1 Detailed Description

Base class which provides an "external" source of postings.
7.51.2 Member Function Documentation

7.51.2.1 virtual bool Xapian::PostingSource::at_end ( ) const  [pure virtual]

Return true if the current position is past the last entry in this list.
At least one of next(), skip_to() or check() will be called before this method is first called.
Implemented in Xapian::FixedWeightPostingSource, and Xapian::ValuePostingSource.

7.51.2.2 virtual bool Xapian::PostingSource::check ( Xapian::docid did, double min_wt )  [virtual]

Check if the specified docid occurs.
The caller is required to ensure that the specified document id did actually exists in the
database. If it does, it must move to that document id, and return true. If it does not, it
may either:

- return true, having moved to a definite position (including "at_end"), which must
  be the same position as skip_to() would have moved to.

or

- return false, having moved to an "indeterminate" position, such that a subsequent
call to next() or skip_to() will move to the next matching position after did.

Generally, this method should act like skip_to() and return true if that can be done at
little extra cost.
Otherwise it should simply check if a particular docid is present, returning true if it is,
and false if it isn’t.
The default implementation calls skip_to() and always returns true.
Xapian will always call init() on a PostingSource before calling this for the first time.
Note: in the case of a multi-database search, the docid specified is the docid in the
single subdatabase relevant to this posting source. See the init() method for details.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td>The document id to check.</td>
</tr>
<tr>
<td>min_wt</td>
<td>The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).</td>
</tr>
</tbody>
</table>

Reimplemented in Xapian::FixedWeightPostingSource, Xapian::DecreasingValue-WeightPostingSource, Xapian::LatLongDistancePostingSource, and Xapian::Value-PostingSource.
7.51.2.3 virtual PostingSource::PostingSource::clone () const
   [virtual]

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie,
the clone does not need to be in the same iteration position as the original: the matcher
will always call init() on the clone before attempting to move the iterator, or read the
information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the Posting-
Source may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". -
If you want to handle the deletion in a special way (for example when wrapping the
Xapian API for use from another language) then you can define a static operator
delete method in your subclass as shown here: http://trac.xapian.-
org/ticket/554#comment:1

Reimplemented in Xapian::FixedWeightPostingSource, Xapian::ValueMapPosting-
Source, Xapian::DecreasingValueWeightPostingSource, Xapian::LatLongDistance-
PostingSource, and Xapian::ValueWeightPostingSource.

7.51.2.4 virtual std::string PostingSource::get_description () const
   [virtual]

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid
forcing those deriving their own PostingSource subclass from having to implement this
(theymay not care what get_description() gives for their subclass).

Reimplemented in Xapian::FixedWeightPostingSource, Xapian::ValueMapPosting-
Source, Xapian::DecreasingValueWeightPostingSource, Xapian::LatLongDistance-
PostingSource, and Xapian::ValueWeightPostingSource.

7.51.2.5 virtual Xapian::docid PostingSource::get_docid () const
   [pure virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document".
See get_weight() for details.

Note: in the case of a multi-database search, the returned docid should be in the single
subdatabase relevant to this posting source. See the init() method for details.

Implemented in Xapian::FixedWeightPostingSource, and Xapian::ValuePostingSource.

---

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.51.2.6 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_est ( )
const [pure virtual]

An estimate of the number of documents this object can return.
It must always be true that:
get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()

Xapian will always call init() on a PostingSource before calling this for the first time.
Implemented in Xapian::FixedWeightPostingSource, and Xapian::ValuePostingSource.

7.51.2.7 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_max ( )
const [pure virtual]

An upper bound on the number of documents this object can return.
Xapian will always call init() on a PostingSource before calling this for the first time.
Implemented in Xapian::FixedWeightPostingSource, and Xapian::ValuePostingSource.

7.51.2.8 virtual Xapian::doccount Xapian::PostingSource::get_termfreq_min ( )
const [pure virtual]

A lower bound on the number of documents this object can return.
Xapian will always call init() on a PostingSource before calling this for the first time.
Implemented in Xapian::FixedWeightPostingSource, and Xapian::ValuePostingSource.

7.51.2.9 virtual double Xapian::PostingSource::get_weight ( ) const [virtual]

Return the weight contribution for the current document.
This default implementation always returns 0, for convenience when implementing
"weight-less" PostingSource subclasses.
This method may assume that it will only be called when there is a "current document".
In detail: Xapian will always call init() on a PostingSource before calling this for the first
time. It will also only call this if the PostingSource reports that it is pointing to a valid
document (ie, it will not call it before calling at least one of next(), skip_to() or check(),
and will ensure that the PostingSource is not at the end by calling at_end()).
Reimplemented in Xapian::FixedWeightPostingSource, Xapian::ValueMapPostingSource,
Xapian::DecreasingValueWeightPostingSource, Xapian::LatLongDistancePostingSource,
and Xapian::ValueWeightPostingSource.

7.51.2.10 virtual void Xapian::PostingSource::init ( const Database & db ) [pure
virtual]

Set this PostingSource to the start of the list of postings.
This is called automatically by the matcher prior to each query being processed.

If a `PostingSource` is used for multiple searches, `init()` will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

**Parameters**

| db | The database which the `PostingSource` should iterate through. |

Note: the database supplied to this method must not be modified: in particular, the `reopen()` method should not be called on it.

Note: in the case of a multi-database search, a separate `PostingSource` will be used for each database (the separate PostingSources will be obtained using `clone()`), and each `PostingSource` will be passed one of the sub-databases as the `db` parameter here. The `db` parameter will therefore always refer to a single database. All docids passed to, or returned from, the `PostingSource` refer to docids in that single database, rather than in the multi-database.

Implemented in `Xapian::FixedWeightPostingSource`, `Xapian::ValueMapPostingSource`, `Xapian::DecreasingValueWeightPostingSource`, `Xapian::LatLongDistancePostingSource`, `Xapian::ValueWeightPostingSource`, and `Xapian::ValuePostingSource`.

### 7.51.2.11 virtual std::string Xapian::PostingSource::name ( ) const [virtual]

Name of the posting source class.

This is used when serialising and unserialising posting sources; for example, for performing remote searches.

If the subclass is in a C++ namespace, the namespace should be included in the name, using `::` as a separator. For example, for a `PostingSource` subclass called `-FooPostingSource` in the “Xapian” namespace the result of this call should be “Xapian::FooPostingSource”.

This should only be implemented if `serialise()` and `unserialise()` are also implemented. The default implementation returns an empty string.

If this returns an empty string, Xapian will assume that `serialise()` and `unserialise()` are not implemented.

Reimplemented in `Xapian::FixedWeightPostingSource`, `Xapian::ValueMapPostingSource`, `Xapian::DecreasingValueWeightPostingSource`, `Xapian::LatLongDistancePostingSource`, and `Xapian::ValueWeightPostingSource`.

### 7.51.2.12 virtual void Xapian::PostingSource::next ( double min_wt ) [pure virtual]

Advance the current position to the next matching document.

The `PostingSource` starts before the first entry in the list, so `next()` must be called before any methods which need the context of the current position.

Xapian will always call `init()` on a `PostingSource` before calling this for the first time.
Parameters

| min_weight | The minimum weight contribution that is needed (this is just a hint which subclasses may ignore). |

Implemented in Xapian::FixedWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::LatLongDistancePostingSource, and Xapian::ValuePostingSource.

7.51.2.13 virtual std::string Xapian::PostingSource::serialise() const [virtual]

Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented in Xapian::FixedWeightPostingSource, Xapian::ValueMapPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::LatLongDistancePostingSource, and Xapian::ValueWeightPostingSource.

7.51.2.14 void Xapian::PostingSource::set_maxweight(double max_weight)

[protected]

Set an upper bound on what get_weight() can return from now on.

This upper bound is used by the matcher to perform various optimisations, so if you can return a good bound, then matches will generally run faster.

This method should be called after calling init(), and may be called during iteration if the upper bound drops.

It is valid for the posting source to have returned a higher value from get_weight() earlier in the iteration, but the posting source must not return a higher value from get_weight() than the currently set upper bound, and the upper bound must not be increased (until init() has been called).

If you don't call this method, the upper bound will default to 0, for convenience when implementing "weight-less" PostingSource subclasses.

Parameters

| max_weight | The upper bound to set. |

7.51.2.15 virtual void Xapian::PostingSource::skip_to(Xapian::docid did, double min_wt) [virtual]

Advance to the specified docid.

If the specified docid isn't in the list, position ourselves on the first document after it (or
at_end() if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified.
A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls next() repeatedly, which works but skip_to() can often be implemented much more efficiently.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the init() method for details.

Parameters

did | The document id to advance to.
min_wt | The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).

Reimplemented in Xapian::FixedWeightPostingSource, Xapian::DecreasingValueWeightPostingSource, Xapian::LatLongDistancePostingSource, and Xapian::ValuePostingSource.

7.51.2.16 virtual PostingSource* Xapian::PostingSource::unserialise ( const std::string & s ) const [virtual]

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters

s | A serialised instance of this PostingSource subclass.

Reimplemented in Xapian::FixedWeightPostingSource, Xapian::ValueMapPostingSource, Xapian::DecreasingValueWeightPostingSource, and Xapian::ValueWeightPostingSource.
virtual PostingSource:: PostingSource::unserialise_with_registry ( const std::string & s, const Registry & registry ) const

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

This method is supplied with a Registry object, which can be used when unserialising objects contained within the posting source. The default implementation simply calls unserialise() which doesn’t take the Registry object, so you do not need to implement this method unless you want to take advantage of the Registry object when unserialising.

Parameters

- `s` A serialised instance of this PostingSource subclass.

Reimplemented in Xapian::LatLongDistancePostingSource.

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

- xapian/postingsource.h

### 7.52 Xapian::Query Class Reference

Class representing a query.

#### Public Types

- `enum op { , OP_ELITE_SET = 10 }`
  
  Query operators.

#### Public Member Functions

- `Query ()`
  
  Default constructor.

- `~Query ()`
  
  Destructor.

- `Query (const Query &o)`
  
  Copying is allowed.

- `Query & operator= (const Query &o)`
  
  Copying is allowed.
7.52.1 Detailed Description

Class representing a query.

7.52.2 Member Enumeration Documentation

7.52.2.1 enum Xapian::Query::op

Query operators.

Enumerator:

**OP_ELITE_SET**  Pick the best N subqueries and combine with OP_OR. If you want to implement a feature which finds documents similar to a piece of text, an obvious approach is to build an "OR" query from all the terms in the text, and run this query against a database containing the documents. However such a query can contain a lots of terms and be quite slow to perform, yet many of these terms don’t contribute usefully to the results.

The OP_ELITE_SET operator can be used instead of OP_OR in this situation. OP_ELITE_SET selects the most important "N" terms and then acts as an O-P_OR query with just these, ignoring any other terms. This will usually return results just as good as the full OP_OR query, but much faster.

In general, the OP_ELITE_SET operator can be used when you have a large OR query, but it doesn’t matter if the search completely ignores some of the less important terms in the query.

The subqueries don’t have to be terms, but if they aren’t then OP_ELITE_SET will look at the estimated frequencies of the subqueries and so could pick a subset which don’t actually match any documents even if the full OR would match some.

You can specify a parameter to the query constructor which control the number of terms which OP_ELITE_SET will pick. If not specified, this defaults to 10 (Xapian used to default to \( \text{ceil}(\sqrt{\text{number\_of\_subqueries}}) \)) if there are more than 100 subqueries, but this rather arbitrary special case was dropped in 1.3.0). For example, this will pick the best 7 terms:

```cpp
Xapian::Query query(Xapian::Query::OP_ELITE_SET, subqs.begin(), subqs.end(), 7);
```

If the number of subqueries is less than this threshold, OP_ELITE_SET behaves identically to OP_OR.
7.52.3 Constructor & Destructor Documentation

7.52.3.1 Xapian::Query::Query ( const Query & o ) [inline]

Copying is allowed.
The internals are reference counted, so copying is cheap.

7.52.3.2 Xapian::Query::Query ( const std::string & term, Xapian::termcount wqf = 1,
                                       Xapian::termpos pos = 0 )

Construct a Query object for a term.

7.52.3.3 Xapian::Query::Query ( Xapian::PostingSource * source )

Construct a Query object for a PostingSource.

7.52.4 Member Function Documentation

7.52.4.1 Query& Xapian::Query::operator= ( const Query & o ) [inline]

Copying is allowed.
The internals are reference counted, so assignment is cheap.
The documentation for this class was generated from the following file:

- xapian/query.h

7.53 Xapian::QueryParser Class Reference

Build a Xapian::Query object from a user query string.

Public Types

- enum feature_flag { FLAG_BOOLEAN = 1, FLAG_PHRASE = 2, FLAG_LOVE-HATE = 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_AUT-O_MULTIWORD_SYNONYMS = 1024, FLAG_DEFAULT = FLAG_PHRASE|FL-A贡_BOOLEAN|FLAG_LOVEHATE }

  Enum of feature flags.

- enum stem_strategy

  Stemming strategies, for use with set_stemming_strategy().
Public Member Functions

- **QueryParser** (const QueryParser &o)
  Copy constructor.
- **QueryParser & operator=** (const QueryParser &o)
  Assignment.
- **QueryParser ()**
  Default constructor.
- **~QueryParser ()**
  Destructor.
- void **set_stemmer** (const Xapian::Stem &stemmer)
  Set the stemmer.
- void **set_stemming_strategy** (stem_strategy strategy)
  Set the stemming strategy.
- void **set_stopper** (const Stopper *stop=NULL)
  Set the stopper.
- void **set_default_op** (Query::op default_op)
  Set the default operator.
- **Query::op get_default_op () const**
  Get the current default operator.
- void **set_database** (const Database &db)
  Specify the database being searched.
- void **set_max_wildcard_expansion** (Xapian::termcount limit)
  Specify the maximum expansion of a wildcard term.
- **Query parse_query** (const std::string &query_string, unsigned flags=FLAG_DEFAULT, const std::string &default_prefix=std::string())
  Parse a query.
- void **add_prefix** (const std::string &field, const std::string &prefix)
  Add a probabilistic term prefix.
- void **add_prefix** (const std::string &field, Xapian::FieldProcessor *proc)
  Register a FieldProcessor.
- void **add_boolean_prefix** (const std::string &field, const std::string &prefix, bool exclusive=true)
  Add a boolean term prefix allowing the user to restrict a search with a boolean filter specified in the free text query.
- void **add_boolean_prefix** (const std::string &field, Xapian::FieldProcessor *proc, bool exclusive=true)
  Register a FieldProcessor for a boolean prefix.
- **TermIterator stoplist_begin () const**
  Iterate over terms omitted from the query as stopwords.
- **TermIterator unstem_begin** (const std::string &term) const
  Iterate over unstemmed forms of the given (stemmed) term used in the query.
- void **add_valuerangeprocessor** (Xapian::ValueRangeProcessor *vrproc)
  Register a ValueRangeProcessor.
7.53 Xapian::QueryParser Class Reference

- std::string get_corrected_query_string () const
  Get the spelling-corrected query string.
- std::string get_description () const
  Return a string describing this object.

7.53.1 Detailed Description

Build a Xapian::Query object from a user query string.

7.53.2 Member Enumeration Documentation

7.53.2.1 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 ALL-CAPS.
- **FLAG_WILDCARD**  Support right truncation (e.g. Xap∗).
  Currently you can’t use wildcards with boolean filter prefixes, or in a phrase (either an explicitly quoted one, or one implicitly generated by hyphens or other punctuation).
  NB: You need to tell the QueryParser object which database to expand wildcards from by calling set_database.
- **FLAG_PURE_NOT**  Allow queries such as ‘NOT apples’. These require the use of a list of all documents in the database which is potentially expensive, so this feature isn’t enabled by default.
- **FLAG_PARTIAL**  Enable partial matching. Partial matching causes the parser to treat the query as a “partially entered” search. This will automatically treat the final word as a wildcarded match, unless it is followed by whitespace, to produce more stable results from interactive searches.
  Currently FLAG_PARTIAL doesn’t do anything if the final word in the query has a boolean filter prefix, or if it is in a phrase (either an explicitly quoted one, or one implicitly generated by hyphens or other punctuation). It also doesn’t do anything if if the final word is part of a value range.
  NB: You need to tell the QueryParser object which database to expand wildcards from by calling set_database.
- **FLAG_SPELLING_CORRECTION**  Enable spelling correction. For each word in the query which doesn’t exist as a term in the database, Database::get_spelling_suggestion() will be called and if a suggestion is returned, a corrected
version of the query string will be built up which can be read using QueryParser::get_corrected_query_string(). The query returned is based on the uncorrected query string however - if you want a parsed query based on the corrected query string, you must call QueryParser::parse_query() again.

NB: You must also call set_database() for this to work.

**FLAG_SYNONYM** Enable synonym operator ‘∼’. NB: You must also call set_database() for this to work.

**FLAG_AUTO_SYNONYMS** Enable automatic use of synonyms for single terms. NB: You must also call set_database() for this to work.

**FLAG_AUTO_MULTIWORD_SYNONYMS** Enable automatic use of synonyms for single terms and groups of terms. NB: You must also call set_database() for this to work.

**FLAG_DEFAULT** The default flags. Used if you don’t explicitly pass any to parse_query(). The default flags are FLAG_PHRASE | FLAG_BOOLEAN | FLAG_LOVEHATE.

Added in Xapian 1.0.11.

### 7.53.3 Member Function Documentation

#### 7.53.3.1 void Xapian::QueryParser::add_boolean_prefix ( const std::string & field, const std::string & prefix, bool exclusive = true )

Add a boolean term prefix allowing the user to restrict a search with a boolean filter specified in the free text query.

For example:

```cpp
goqp.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 |
| exclusive | If true, each document can have at most one term with this prefix, so multiple filters with this prefix should be combined with OP_OR. If false, each document can have multiple terms with this prefix, so multiple filters should be combined with OP_AND, like happens with filters with different prefixes. [default: true] |

7.53.3.2 void Xapian::QueryParser::add_boolean_prefix ( const std::string & field, Xapian::FieldProcessor * proc, bool exclusive = true )

Register a FieldProcessor for a boolean prefix.

Experimental API - may change.

7.53.3.3 void Xapian::QueryParser::add_prefix ( const std::string & field, const std::string & prefix )

Add a probabilistic term prefix.

For example:

```cpp
qp.add_prefix("author", "A");
```

This allows the user to search for author:Orwell which will be converted to a search for the term "Aorwell".

Multiple fields can be mapped to the same prefix. For example, you can make title: and subject: aliases for each other.

As of 1.0.4, you can call this method multiple times with the same value of field to allow a single field to be mapped to multiple prefixes. Multiple terms being generated for such a field, and combined with Xapian::Query::OP_OR.

If any prefixes are specified for the empty field name (i.e. you call this method with an empty string as the first parameter) these prefixes will be used for terms without a field specifier. If you do this and also specify the default_prefix parameter to parse_query(), then the default_prefix parameter will override.

If the prefix parameter is empty, then "field:word" will produce the term "word" (and this can be one of several prefixes for a particular field, or for terms without a field specifier).

If you call add_prefix() and add_boolean_prefix() for the same value of field, a Xapian::InvalidOperationError exception will be thrown.
In 1.0.3 and earlier, subsequent calls to this method with the same value of field had no effect.

**Parameters**

<table>
<thead>
<tr>
<th>field</th>
<th>The user visible field name</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix</td>
<td>The term prefix to map this to</td>
</tr>
</tbody>
</table>

7.53.3.4 void Xapian::QueryParser::add_prefix ( const std::string & field,
Xapian::FieldProcessor ∗ proc )

Register a FieldProcessor.

Experimental API - may change.

7.53.3.5 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.

7.53.3.6 Query::op Xapian::QueryParser::get_default_op ( ) const

Get the current default operator.

7.53.3.7 Query Xapian::QueryParser::parse_query ( const std::string & query_string,
unsigned flags = FLAG_DEFAULT, const std::string & default_prefix =
std::string() )

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 FLAG_DEFAULT). |
| default_prefix | The default term prefix to use (default none). For example, you can pass "A" when parsing an "Author" field. |

**Exceptions**

If the query string can’t be parsed, then Xapian::QueryParserError is thrown. You can get an English error message to report to the user by catching it and calling get_msg() on the caught exception. The current possible values (in case you want to translate them) are:
- Unknown range operation
- parse error
- Syntax: `<expression> AND <expression>`
- Syntax: `<expression> AND NOT <expression>`
- Syntax: `<expression> NOT <expression>`
- Syntax: `<expression> OR <expression>`
- Syntax: `<expression> XOR <expression>`

### 7.53.3.8 void Xapian::QueryParser::set_database (const Database & db)

Specify the database being searched.

**Parameters**

| db       | The database to use for wildcard expansion (FLAG_WILDCARD and FLAG_PARTIAL), spelling correction (FLAG_SPELLING_CORRECTION), and synonyms (FLAG_SYNONYM, FLAG_AUTO_SYNONYMS, and FLAG_AUTO_MULTIWORD_SYNONYMS). |

### 7.53.3.9 void Xapian::QueryParser::set_default_op (Query::op default_op)

Set the default operator.

**Parameters**

| default_op | The operator to use to combine non-filter query items when no explicit operator is used. |

So for example, ‘weather forecast’ is parsed as if it were ‘weather OR forecast’ by default.

The most useful values for this are OP_OR (the default) and OP_AND, OP_NEAR, OP_PHRASE, OP_ELITE_SET and OP_SYNONYM are also permitted. Passing other values will result in InvalidArgumentError being thrown.

### 7.53.3.10 void Xapian::QueryParser::set_max_wildcard_expansion (Xapian::termcount limit)

Specify the maximum expansion of a wildcard term.

Note: you must also set FLAG_WILDCARD for wildcard expansion to happen.
Parameters

| limit | The maximum number of terms each wildcard in the query can expand to, or 0 for no limit (which is the default). |

### 7.53.3.11 void Xapian::QueryParser::set_stemmer ( const Xapian::Stem & stemmer )

Set the stemmer.

This sets the stemming algorithm which will be used by the query parser. The stemming algorithm will be used according to the stemming strategy set by `set_stemming_strategy()`. As of 1.3.1, this defaults to STEM_SOME, but in earlier versions the default was STEM_NONE. If you want to work with older versions, you should explicitly set a stemming strategy as well as setting a stemmer, otherwise your stemmer won’t actually be used.

Parameters

| stemmer | The Xapian::Stem object to set. |

### 7.53.3.12 void Xapian::QueryParser::set_stemming_strategy ( stem_strategy strategy )

Set the stemming strategy.

This controls how the query parser will apply the stemming algorithm. Note that the stemming algorithm is only applied to words in probabilistic fields - boolean filter terms are never stemmed.

Parameters

<table>
<thead>
<tr>
<th>strategy</th>
<th>The strategy to use - possible values are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• STEM_NONE: Don’t perform any stemming. (default in Xapian &lt;= 1.3.0)</td>
<td></td>
</tr>
<tr>
<td>• STEM_SOME: Search for stemmed forms of terms except for those which start with a capital letter, or are followed by certain characters (currently: /[@&lt;&gt;]=[^{ ]}, or are used with operators which need positional information. Stemmed terms are prefixed with 'Z'. (default in Xapian &gt;= 1.3.1)</td>
<td></td>
</tr>
<tr>
<td>• STEM_ALL: Search for stemmed forms of all words (note: no 'Z' prefix is added).</td>
<td></td>
</tr>
<tr>
<td>• STEM_ALL_Z: Search for stemmed forms of all words (note: 'Z' prefix is added). (new in Xapian 1.2.11 and 1.3.1)</td>
<td></td>
</tr>
</tbody>
</table>
7.53.3.13  void Xapian::QueryParser::set_stopper ( const Stopper * stop = NULL )
Set the stopper.

Parameters

| stop | The Stopper object to set (default NULL, which means no stopwords). |

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

- xapian/queryparser.h

7.54  Xapian::QueryParserError Class Reference

Indicates a query string can’t be parsed.

Inheritance diagram for Xapian::QueryParserError:

```
Xapian::QueryParserError
|        |
|        |
|        |
```

Public Member Functions

- **QueryParserError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)

  General purpose constructor.

- **QueryParserError** (const std::string &msg_ int errno_)

  Construct from message and errno value.
7.54.1 Detailed Description

Indicates a query string can’t be parsed.

7.54.2 Constructor & Destructor Documentation

7.54.2.1 Xapian::QueryParserError::QueryParserError ( const std::string & msg_, const std::string & context_ = std::string(), int errno_ = 0 ) [inline, explicit]

General purpose constructor.

Parameters

- **msg_**: Message giving details of the error, intended for human consumption.
- **context_**: Optional context information for this error.
- **errno_**: Optional errno value associated with this error.

7.54.2.2 Xapian::QueryParserError::QueryParserError ( const std::string & msg_, int errno_ ) [inline]

Construct from message and errno value.

Parameters

- **msg_**: Message giving details of the error, intended for human consumption.
- **errno_**: Optional errno value associated with this error.

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

- xapian/error.h

7.55 Xapian::RangeError Class Reference

**RangeError** indicates an attempt to access outside the bounds of a container.
Inheritance diagram for Xapian::RangeError:

![Inheritance Diagram](image)

**Public Member Functions**

- **RangeError** (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)
  
  *General purpose constructor.*

- **RangeError** (const std::string &msg_, int errno_)
  
  *Construct from message and errno value.*

**7.55.1 Detailed Description**

*RangeError* indicates an attempt to access outside the bounds of a container.

**7.55.2 Constructor & Destructor Documentation**

**7.55.2.1 Xapian::RangeError::RangeError** (const std::string &msg_, const std::string &context_, int errno_ = 0) [inline, explicit]

*General purpose constructor.*

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg_</td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>
7.55.2.2 Xapian::RangeError::RangeError ( const std::string & msg_, int errno_ )

[inline]

Construct from message and errno value.

Parameters

- **msg_**: Message giving details of the error, intended for human consumption.
- **errno_**: Optional errno value associated with this error.

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

- `xapian/error.h`

### 7.56 Xapian::Registry Class Reference

Registry for user subclasses.

#### Public Member Functions

- **Registry** (const Registry &other)
  
  Copy constructor.
- **Registry & operator=** (const Registry &other)
  
  Assignment operator.
- **Registry ()**
  
  Default constructor.
- **void register_weighting_scheme** (const Xapian::Weight &wt)
  
  Register a weighting scheme.
- **const Xapian::Weight * get_weighting_scheme** (const std::string &name) const
  
  Get the weighting scheme given a name.
- **void register_posting_source** (const Xapian::PostingSource &source)
  
  Register a user-defined posting source class.
- **const Xapian::PostingSource * get_posting_source** (const std::string &name) const
  
  Get a posting source given a name.
- **void register_match_spy** (const Xapian::MatchSpy &spy)
  
  Register a user-defined match spy class.
- **const Xapian::MatchSpy * get_match_spy** (const std::string &name) const
  
  Get a match spy given a name.
- **void register_lat_long_metric** (const Xapian::LatLongMetric &metric)
  
  Register a user-defined lat-long metric class.
- **const Xapian::LatLongMetric * get_lat_long_metric** (const std::string &name) const
  
  Get a lat-long metric given a name.
7.56 Xapian::Registry Class Reference

7.56.1 Detailed Description

Registry for user subclasses.

This class provides a way for the remote server to look up user subclasses when unserialising.

7.56.2 Constructor & Destructor Documentation

7.56.2.1 Xapian::Registry::Registry (const Registry &other)

Copy constructor.
The internals are reference counted, so copying is cheap.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
</tr>
</tbody>
</table>

7.56.2.2 Xapian::Registry::Registry()

Default constructor.
The registry will contain all standard subclasses of user-subclassable classes.

7.56.3 Member Function Documentation

7.56.3.1 const Xapian::LatLongMetric* Xapian::Registry::get_lat_long_metric (const std::string &name) const

Get a lat-long metric given a name.
The returned metric is owned by the registry object.
Returns NULL if the metric could not be found.

7.56.3.2 const Xapian::MatchSpy* Xapian::Registry::get_match_spy (const std::string &name) const

Get a match spy given a name.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
</tbody>
</table>
Returns
An object with the requested name, or NULL if the match spy could not be found. The returned object is owned by the registry and so must not be deleted by the caller.

7.56.3.3 const Xapian::PostingSource* Xapian::Registry::get_posting_source (const std::string & name) const

Get a posting source given a name.

Parameters

| name    | The name of the posting source to find. |

Returns
An object with the requested name, or NULL if the posting source could not be found. The returned object is owned by the registry and so must not be deleted by the caller.

7.56.3.4 const Xapian::Weight* Xapian::Registry::get_weighting_scheme (const std::string & name) const

Get the weighting scheme given a name.

Parameters

| name    | The name of the weighting scheme to find. |

Returns
An object with the requested name, or NULL if the weighting scheme could not be found. The returned object is owned by the registry and so must not be deleted by the caller.

7.56.3.5 Registry& Xapian::Registry::operator= (const Registry & other)

Assignment operator.
The internals are reference counted, so assignment is cheap.

Parameters

| other   | The object to copy. |
### 7.57 Xapian::RSet Class Reference

#### 7.56.3.6 void Xapian::Registry::register_match_spy ( const Xapian::MatchSpy & spy )

Register a user-defined match spy class.

**Parameters**

| spy | The match spy to register. |

#### 7.56.3.7 void Xapian::Registry::register_posting_source ( const Xapian::PostingSource & source )

Register a user-defined posting source class.

**Parameters**

| source | The posting source to register. |

#### 7.56.3.8 void Xapian::Registry::register_weighting_scheme ( const Xapian::Weight & wt )

Register a weighting scheme.

**Parameters**

| wt | The weighting scheme to register. |

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

- xapian/registry.h

### 7.57 Xapian::RSet Class Reference

A relevance set (R-Set).

#### Public Member Functions

- **RSet (const RSet &rset)**
  
  *Copy constructor.*

- **void operator= (const RSet &rset)**
  
  *Assignment operator.*

- **RSet ()**
  
  *Default constructor.*

- **~RSet ()**
Destructor.

- `Xapian::doccount size () const`
  The number of documents in this R-Set.

- `bool empty () const`
  Test if this R-Set is empty.

- `void add_document (Xapian::docid did)`
  Add a document to the relevance set.

- `void add_document (const Xapian::MSetIterator &i)`
  Add a document to the relevance set.

- `void remove_document (Xapian::docid did)`
  Remove a document from the relevance set.

- `void remove_document (const Xapian::MSetIterator &i)`
  Remove a document from the relevance set.

- `bool contains (Xapian::docid did) const`
  Test if a given document is in the relevance set.

- `bool contains (const Xapian::MSetIterator &i) const`
  Test if a given document is in the relevance set.

- `std::string get_description () const`
  Return a string describing this object.

### 7.57.1 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.

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

- `xapian/enquire.h`

### 7.58 Xapian::RuntimeError Class Reference

The base class for exceptions indicating errors only detectable at runtime.
7.58.1 Detailed Description

The base class for exceptions indicating errors only detectable at runtime.

A subclass of `RuntimeError` will be thrown if Xapian detects an error which is exception derived from `RuntimeError` is thrown when an error is caused by problems with the data or environment rather than a programming mistake.

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

- xapian/error.h

7.59 Xapian::SerialisationError Class Reference

Indicates an error in the std::string serialisation of an object.
Inheritance diagram for Xapian::SerialisationError:

```
Xapian::Error

Xapian::RuntimeError

Xapian::SerialisationError
```

Public Member Functions

- `SerialisationError (const std::string &msg_, const std::string &context_=std::string(), int errno_=0)`
  
  General purpose constructor.

- `SerialisationError (const std::string &msg_, int errno_)`
  
  Construct from message and errno value.

7.59.1 Detailed Description

Indicates an error in the std::string serialisation of an object.

7.59.2 Constructor & Destructor Documentation

7.59.2.1 Xapian::SerialisationError::SerialisationError ( const std::string & msg_, const std::string & context_ = std::string(), int errno_ = 0 ) [inline, explicit]

General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>msg_</th>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>context_</td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>
7.60 Xapian::SimpleStopper Class Reference

Simple implementation of Stopper class - this will suit most users.

Inheritance diagram for Xapian::SimpleStopper:

```
    Xapian::SimpleStopper
        Xapian::Stopper
```

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 std::string get_description () const
  
  Return a string describing this object.

7.60.1 Detailed Description

Simple implementation of Stopper class - this will suit most users.

7.60.2 Member Function Documentation

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

Is term a stop-word?

Parameters

| term | The term to test. |

Implements Xapian::Stopper.

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

• xapian/queryparser.h

7.61 Xapian::Stem Class Reference

Class representing a stemming algorithm.

Public Member Functions

• Stem (const Stem &o)
  
  Copy constructor.
• Stem & 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 (StemImplementation *p)
  
  Construct a Xapian::Stem object with a user-provided stemming algorithm.
• ~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.

### 7.61.1 Detailed Description

Class representing a stemming algorithm.

### 7.61.2 Constructor & Destructor Documentation

#### 7.61.2.1 Xapian::Stem::Stem ()

Construct a Xapian::Stem object which doesn’t change terms.
Equivalent to Stem("none").

#### 7.61.2.2 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
- armenian (hy)
- basque (eu)
- catalan (ca)
- danish (da)
- dutch (nl)
- english (en) - Martin Porter’s 2002 revision of his stemmer
- english_lovins (lovins) - Lovin’s stemmer
- english_porter (porter) - Porter’s stemmer as described in his 1980 paper
• finnish (fi)
• french (fr)
• german (de)
• german2 - Normalises umlauts and ß
• hungarian (hu)
• italian (it)
• kraaij_pohlmann - A different Dutch stemmer
• norwegian (nb, nn, no)
• portuguese (pt)
• romanian (ro)
• russian (ru)
• spanish (es)
• swedish (sv)
• turkish (tr)

Exceptions

`Xapian::InvalidArgumentError` is thrown if language isn’t recognised.

7.61.2.3 Xapian::Stem::Stem (StemImplementation * p) [explicit]

Construct a Xapian::Stem object with a user-provided stemming algorithm.

You can subclass Xapian::StemImplementation to implement your own stemming algorithm (or to wrap a third-party algorithm) and then wrap your implementation in a Xapian::Stem object to pass to the Xapian API.

Parameters

- `p` The user-subclassed StemImplementation object. This is reference counted, and so will be automatically deleted by the Xapian::Stem wrapper when no longer required.

7.61.3 Member Function Documentation

7.61.3.1 static std::string Xapian::Stem::get_available_languages ( ) [static]

Return a list of available languages.
Each stemmer is only included once in the list (not once for each alias). The name
included is the English name of the language.

The list is returned as a string, with language names separated by spaces. This is a
static method, so a Xapian::Stem object is not required for this operation.

7.61.3.2  std::string Xapian::Stem::operator() ( const std::string & word ) const

Stem a word.

Parameters

    word  a word to stem.

Returns

    the stem

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

     • xapian/stem.h

7.62  Xapian::StemImplementation Struct Reference

Class representing a stemming algorithm implementation.

Public Member Functions

• virtual ~StemImplementation ()

    Virtual destructor.

• virtual std::string operator() (const std::string &word)=0

    Stem the specified word.

• virtual std::string get_description () const =0

    Return a string describing this object.

7.62.1  Detailed Description

Class representing a stemming algorithm implementation.

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

     • xapian/stem.h
7.63  Xapian::Stopper Class Reference

Base class for stop-word decision functor.
Inheritance diagram for Xapian::Stopper:

```
Xapian::Stopper
│
│ Xapian::SimpleStopper
```

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.*

7.63.1  Detailed Description

Base class for stop-word decision functor.

7.63.2  Member Function Documentation

7.63.2.1  virtual bool Xapian::Stopper::operator() ( const std::string & term ) const  [pure

Is term a stop-word?

Parameters

| term | The term to test. |

Implemented in Xapian::SimpleStopper.
7.64 Xapian::StringValueRangeProcessor Class Reference

Handle a string range.

Inheritance diagram for Xapian::StringValueRangeProcessor:

```
Xapian::ValueRangeProcessor
    Xapian::StringValueRangeProcessor
        Xapian::DateValueRangeProcessor
        Xapian::NumberValueRangeProcessor
```

Public Member Functions

- **StringValueRangeProcessor (Xapian::valueno slot_)**
  Constructor.
- **StringValueRangeProcessor (Xapian::valueno slot_, const std::string &str_, bool prefix_=true)**
  Constructor.
- **Xapian::valueno operator() (std::string &begin, std::string &end)**
  Check for a valid string range.

7.64.1 Detailed Description

Handle a string range.

The end points can be any strings.

7.64.2 Constructor & Destructor Documentation

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

- xapian/queryparser.h
7.64.2.1 Xapian::StringValueRangeProcessor::StringValueRangeProcessor ( Xapian::valueno slot_ ) [inline]

 Constructor.

 Parameters

| slot_ | The value number to return from operator(). |

7.64.2.2 Xapian::StringValueRangeProcessor::StringValueRangeProcessor ( Xapian::valueno slot_, const std::string & str_, bool prefix_ = true ) [inline]

 Constructor.

 Parameters

| slot_ | The value number to return from operator(). |
| str_  | A string to look for to recognise values as belonging to this range. |
| prefix_ | Flag specifying whether to check for str_ as a prefix or a suffix. |

7.64.3 Member Function Documentation

7.64.3.1 Xapian::valueno Xapian::StringValueRangeProcessor::operator() ( std::string & begin, std::string & end ) [virtual]

 Check for a valid string range.

 Parameters

| in,out | begin | The start of the range as specified in the query string by the user. This parameter is a non-const reference so the ValueRangeProcessor can modify it to return the value to start the range with. |
| in,out | end | The end of the range. This is also a non-const reference so it can be modified. |

 Returns

 A StringValueRangeProcessor always accepts a range it is offered, and returns the value of slot_ passed at construction time. It doesn’t modify begin or end.

 Implements Xapian::ValueRangeProcessor.

 Reimplemented in Xapian::NumberValueRangeProcessor, and Xapian::DateValueRangeProcessor.

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

 • xapian/queryparser.h
7.65  Xapian::TermGenerator Class Reference

Parses a piece of text and generate terms.

Public Types

- enum flags { FLAG_SPELLING = 128 }
  
  Flags to OR together and pass to TermGenerator::set_flags().

- enum stem_strategy
  
  Stemming strategies, for use with set_stemming_strategy().

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 set_stemming_strategy (stem_strategy strategy)
  
  Set the stemming strategy.

- void set_max_word_length (unsigned max_word_length)
  
  Set the maximum length word to index.

- void index_text (const Xapian::Utf8Iterator &itor, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())
  
  Index some text.

- void index_text (const std::string &text, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())
  
  Index some text in a std::string.
• void index_text_without_positions (const Xapian::Utf8Iterator &itor, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())
  Index some text without positional information.
• void index_text_without_positions (const std::string &text, Xapian::termcount wdf_inc=1, const std::string &prefix=std::string())
  Index some text in a std::string without positional information.
• void increase_termpos (Xapian::termcount delta=100)
  Increase the term position used by index_text.
• 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.

7.65.1 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.

7.65.2 Member Enumeration Documentation

7.65.2.1 enum Xapian::TermGenerator::flags

Flags to OR together and pass to TermGenerator::set_flags().

Enumerator:

FLAG_SPELLING Index data required for spelling correction.

7.65.3 Member Function Documentation

7.65.3.1 void Xapian::TermGenerator::increase_termpos ( Xapian::termcount delta
  = 100 )

Increase the term position used by index_text.

This can be used between indexing text from different fields or other places to prevent phrase searches from spanning between them (e.g. between the title and body text, or between two chapters in a book).

Parameters

| delta | Amount to increase the term position by (default: 100). |
7.65.3.2 void Xapian::TermGenerator::index_text ( const Xapian::Utf8Iterator & itor, Xapian::termcount wdf_inc = 1, const std::string & prefix = std::string() )

Index some text.

Parameters

<table>
<thead>
<tr>
<th>itor</th>
<th>Utf8Iterator pointing to the text to index.</th>
</tr>
</thead>
<tbody>
<tr>
<td>wdf_inc</td>
<td>The wdf increment (default 1).</td>
</tr>
<tr>
<td>prefix</td>
<td>The term prefix to use (default is no prefix).</td>
</tr>
</tbody>
</table>

7.65.3.3 void Xapian::TermGenerator::index_text ( const std::string & text, Xapian::termcount wdf_inc = 1, const std::string & prefix = std::string() ) [inline]

Index some text in a std::string.

Parameters

<table>
<thead>
<tr>
<th>text</th>
<th>The text to index.</th>
</tr>
</thead>
<tbody>
<tr>
<td>wdf_inc</td>
<td>The wdf increment (default 1).</td>
</tr>
<tr>
<td>prefix</td>
<td>The term prefix to use (default is no prefix).</td>
</tr>
</tbody>
</table>

7.65.3.4 void Xapian::TermGenerator::index_text_without_positions ( const Xapian::Utf8Iterator & itor, Xapian::termcount wdf_inc = 1, const std::string & prefix = std::string() )

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.

Parameters

<table>
<thead>
<tr>
<th>itor</th>
<th>Utf8Iterator pointing to the text to index.</th>
</tr>
</thead>
<tbody>
<tr>
<td>wdf_inc</td>
<td>The wdf increment (default 1).</td>
</tr>
<tr>
<td>prefix</td>
<td>The term prefix to use (default is no prefix).</td>
</tr>
</tbody>
</table>

7.65.3.5 void Xapian::TermGenerator::index_text_without_positions ( const std::string & text, Xapian::termcount wdf_inc = 1, const std::string & prefix = std::string() ) [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.
supported.

Parameters

+---+--------------------------------------------------+
|    | The text to index.                             |
| text|                                                 |
+---+--------------------------------------------------+
| wdf_inc | The wdf increment (default 1).               |
+---+--------------------------------------------------+
| prefix | The term prefix to use (default is no prefix). |
+---+--------------------------------------------------+

7.65.3.6 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

+---+--------------------------------------------------+
|    | Flags to XOR.                                  |
| toggle|                                                |
+---+--------------------------------------------------+
| mask | Flags to AND with first.                       |
+---+--------------------------------------------------+

Returns

The old flags setting.

7.65.3.7 void Xapian::TermGenerator::set_max_word_length ( unsigned max_word_length )

Set the maximum length word to index.

The limit is on the length of a word prior to stemming and prior to adding any term prefix.

The backends mostly impose a limit on the length of terms (often of about 240 bytes),
but it's generally useful to have a lower limit to help prevent the index being bloated
by useless junk terms from trying to indexing things like binary data, uuencoded data,
ASCII art, etc.

This method was new in Xapian 1.3.1.

Parameters

+---+--------------------------------------------------------------------------------------+
| max_word_length | The maximum length word to index, in bytes in UTF-8 representation. Default is 64. |
+------------------+--------------------------------------------------------------------------------------+

7.65.3.8 void Xapian::TermGenerator::set_stemming_strategy ( stem_strategy strategy )

Set the stemming strategy.
This method controls how the stemming algorithm is applied. It was new in Xapian 1.3.1.

Parameters

<table>
<thead>
<tr>
<th>strategy</th>
<th>The strategy to use - possible values are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEM_NONE</td>
<td>Don’t perform any stemming - only unstemmed terms are generated.</td>
</tr>
<tr>
<td>STEM_SOME</td>
<td>Generate both stemmed (with a “Z” prefix) and unstemmed terms. This is the default strategy.</td>
</tr>
<tr>
<td>STEM_ALL</td>
<td>Generate only stemmed terms (but without a “Z” prefix).</td>
</tr>
<tr>
<td>STEM_ALL_Z</td>
<td>Generate only stemmed terms (with a “Z” prefix).</td>
</tr>
</tbody>
</table>

```
7.65.3.9 void Xapian::TermGenerator::set_stopper ( const Xapian::Stopper ∗ stop = NULL )
```

Set the Xapian::Stopper object to be used for identifying stopwords.

Stemmed forms of stopwords aren’t indexed, but unstemmed forms still are so that searches for phrases including stop words still work.

Parameters

| stop | The Stopper object to set (default NULL, which means no stopwords). |

```
7.65.3.10 void Xapian::TermGenerator::set_termpos ( Xapian::termcount termpos )
```

Set the current term position.

Parameters

| termpos | The new term position to set. |

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

- xapian/termgenerator.h
Public Member Functions

- **TermIterator (const TermIterator &o)**
  
  Copy constructor.

- **TermIterator & operator= (const TermIterator &o)**
  
  Assignment.

- **TermIterator ()**
  
  Default constructor.

- **~TermIterator ()**
  
  Destructor.

- **std::string operator* () const**
  
  Return the term at the current position.

- **Xapian::termcount get_wdf () const**
  
  Return the wdf for the term at the current position.

- **Xapian::doccount get_termfreq () const**
  
  Return the term frequency for the term at the current position.

- **Xapian::termcount positionlist_count () const**
  
  Return the length of the position list for the current position.

- **PositionIterator positionlist_begin () const**
  
  Return a PositionIterator for the current term.

- **PositionIterator positionlist_end () const**
  
  Return an end PositionIterator for the current term.

- **TermIterator & operator++ ()**
  
  Advance the iterator to the next position.

- **DerefWrapper_<std::string> operator++ (int)**
  
  Advance the iterator to the next position (postfix version).

- **void skip_to (const std::string &term)**
  
  Advance the iterator to term term.

- **std::string get_description () const**
  
  Return a string describing this object.

### 7.66.1 Detailed Description

Class for iterating over a list of terms.

### 7.66.2 Constructor & Destructor Documentation

#### 7.66.2.1 Xapian::TermIterator::TermIterator() [inline]

Default constructor.

Creates an uninitialised iterator, which can’t be used before being assigned to, but is sometimes syntactically convenient.
7.66.3 Member Function Documentation

7.66.3.1 void Xapian::TermIterator::skip_to (const std::string & term)

Advance the iterator to term term.

If the iteration is over an unsorted list of terms, then this method will throw Xapian::InvalidOperationError.

Parameters

| term | The term to advance to. If this term isn’t in the stream being iterated, then the iterator is moved to the next term after it which is. |

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

- xapian/termiterator.h

7.67 Xapian::TfIdfWeight Class Reference

Xapian::Weight subclass implementing the tf-idf weighting scheme.

Inheritance diagram for Xapian::TfIdfWeight:

```
Xapian::Weight
    |
    v
Xapian::TfIdfWeight
```

Public Member Functions

- **TfIdfWeight** (const std::string &normalizations)
  
  Construct a TfIdfWeight.

- **std::string name () const**
  
  Return the name of this weighting scheme.

- **std::string serialise () const**
  
  Return this object’s parameters serialised as a single string.
Class Documentation

- `TfIdfWeight * unserialise (const std::string &s) const`
  Unserialise parameters.

- `double get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const`
  Calculate the weight contribution for this object's term to a document.

- `double get_maxpart () const`
  Return an upper bound on what `get_sumpart()` can return for any document.

- `double get_sumextra (Xapian::termcount doclen) const`
  Calculate the term-independent weight component for a document.

- `double get_maxextra () const`
  Return an upper bound on what `get_sumextra()` can return for any document.

7.67.1 Detailed Description

`Xapian::Weight` subclass implementing the tf-idf weighting scheme.

7.67.2 Constructor & Destructor Documentation

7.67.2.1 `Xapian::TfIdfWeight::TfIdfWeight ( const std::string & normalizations ) [explicit]`

Construct a `TfIdfWeight`.

Parameters

| normalizations | A three character string indicating the normalizations to be used for the tf(wdf), idf and document weight respectively. |

The first character specifies the normalization for the wdf for which the following normalizations are currently available:

'\n':None. \( \text{wdf}_n = \text{wdf} \)
'b':Boolean \( \text{wdf}_n = 1 \) if term in document else \( \text{wdf}_n = 0 \)
's':Square \( \text{wdf}_n = \text{wdf} \times \text{wdf} \)
'l':Logarithmic \( \text{wdf}_n = 1 + \log_e (\text{wdf}) \)

The Max-wdf and Augmented Max wdf normalizations aren't yet implemented.

The second character indicates the normalization for the idf, the following of which are currently available:

'\n':None idf=1
't':Tfidf \( \text{idf}_n = \log(N/\text{Termfreq}) \) where \( N \) is the number of documents in collection and Termfreq is the number of documents which are indexed by the term t.
'p':Prob \( \text{idf}_n = \log((N-\text{Termfreq})/\text{Termfreq}) \)

The third and the final character indicates the normalization for the document weight of which the following are currently available:

'\n':None wtn=fn-idfn
Implementing more normalizations for the weight requires access to statistics such as the weight of all terms in the document indexed by the term in the query. This is not available from the current backend.
More normalizations for all components can be implemented by changing the backend to acquire the statistics required for the normalizations which are not currently available from Xapian::Weight.

The default string is "ntn".

7.67.3 Member Function Documentation

7.67.3.1 double Xapian::TfidfWeight::get_maxextra( ) const [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.67.3.2 double Xapian::TfidfWeight::get_maxpart( ) const [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.67.3.3 double Xapian::TfidfWeight::get_sumextra( Xapian::termcount doclen ) const [virtual]

Calculate the term-independent weight component for a document.

The parameter gives information about the document which may be used in the calculations:

Parameters

| doclen | The document's length (unnormalised). |

Implements Xapian::Weight.

7.67.3.4 double Xapian::TfidfWeight::get_sumpart( Xapian::termcount wdf,
Xapian::termcount doclen ) const [virtual]

Calculate the weight contribution for this object's term to a document.

The parameters give information about the document which may be used in the calculations:
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wdf</code></td>
<td>The within document frequency of the term in the document.</td>
</tr>
<tr>
<td><code>doclen</code></td>
<td>The document's length (unnormalised).</td>
</tr>
</tbody>
</table>

Implements `Xapian::Weight`.

7.6.7.3.5 `std::string` `Xapian::TfidfWeight::name ( ) const` [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called `FooWeight`, return "FooWeight" from this method (`Xapian::BM25Weight` returns "Xapian::BM25Weight" here).

If you don’t want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented from `Xapian::Weight`.

7.6.7.3.6 `std::string` `Xapian::TfidfWeight::serialise ( ) const` [virtual]

Return this object’s parameters serialised as a single string.

If you don’t want to support the remote backend, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Reimplemented from `Xapian::Weight`.

7.6.7.3.7 `TfidfWeight` `Xapian::TfidfWeight::unserialise ( const std::string & s ) const` [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the `serialise()` method and allocates and returns a new object initialised with them.

If you don’t want to support the remote backend, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Note that the returned object will be deallocated by `Xapian` after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the `Xapian` API for use from another language) then you can define a static operator delete method in your subclass as shown here: [http://trac.xapian.org/ticket/554#comment:1](http://trac.xapian.org/ticket/554#comment:1)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>s</code></td>
<td>A string containing the serialised parameters.</td>
</tr>
</tbody>
</table>
Reimplemented from \texttt{Xapian::Weight}.

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

- \texttt{xapian/weight.h}

### 7.68 Xapian::TradWeight Class Reference

\texttt{Xapian::Weight} subclass implementing the traditional probabilistic formula.

Inheritance diagram for \texttt{Xapian::TradWeight}:

```
Xapian::Weight
   \arrow[blue]{up}
   \arrow[blue]{down}
Xapian::TradWeight
```

### Public Member Functions

- \texttt{TradWeight (double k=1.0)}
  
  Construct a \texttt{TradWeight}.

- \texttt{std::string name () const}
  
  Return the name of this weighting scheme.

- \texttt{std::string serialise () const}
  
  Return this object's parameters serialised as a single string.

- \texttt{TradWeight \* unserialise (const std::string \&s) const}
  
  Unserialise parameters.

- \texttt{double get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const}
  
  Calculate the weight contribution for this object's term to a document.

- \texttt{double get_maxpart () const}
  
  Return an upper bound on what \texttt{get_sumpart()} can return for any document.

- \texttt{double get_sumextra (Xapian::termcount doclen) const}
  
  Calculate the term-independent weight component for a document.

- \texttt{double get_maxextra () const}
  
  Return an upper bound on what \texttt{get_sumextra()} can return for any document.
7.68.1 Detailed Description

**Xapian::Weight** subclass implementing the traditional probabilistic formula.

This class implements the “traditional” Probabilistic Weighting scheme, as described by the early papers on Probabilistic Retrieval. BM25 generally gives better results.

TradWeight(k) is equivalent to BM25Weight(k, 0, 0, 1, 0), except that the latter returns weights (k+1) times larger.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 **Xapian::TradWeight::TradWeight** ( double k = 1.0 ) [inline, explicit]

Construct a TradWeight.

Parameters

| k       | A non-negative parameter controlling how influential within-document-frequency (wdf) and document length are. k=0 means that wdf and document length don’t affect the weights. The larger k is, the more they do. (default 1) |

7.68.3 Member Function Documentation

7.68.3.1 **double Xapian::TradWeight::get_maxextra** ( ) const [virtual]

Return an upper bound on what get_sumextra() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.68.3.2 **double Xapian::TradWeight::get_maxpart** ( ) const [virtual]

Return an upper bound on what get_sumpart() can return for any document.

This information is used by the matcher to perform various optimisations, so strive to make the bound as tight as possible.

Implements Xapian::Weight.

7.68.3.3 **double Xapian::TradWeight::get_sumextra** ( Xapian::termcount doclen ) const [virtual]

Calculate the term-independent weight component for a document.
The parameter gives information about the document which may be used in the calculations:

Parameters

- **doclen**: The document’s length (unnormalised).

Implements Xapian::Weight.

### 7.68.3.4 double Xapian::TradWeight::get_sumpart ( Xapian::termcount wdf, Xapian::termcount doclen ) const [virtual]

Calculate the weight contribution for this object’s term to a document.

The parameters give information about the document which may be used in the calculations:

Parameters

- **wdf**: The within document frequency of the term in the document.
- **doclen**: The document’s length (unnormalised).

Implements Xapian::Weight.

### 7.68.3.5 std::string Xapian::TradWeight::name ( ) const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called FooWeight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don’t want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented from Xapian::Weight.

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

Return this object’s parameters serialised as a single string.

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::Weight.
Unserialise parameters.

This method unserialises parameters serialised by the `serialise()` method and allocates and returns a new object initialised with them.

If you don’t want to support the remote backend, you can use the default implementation which simply throws `Xapian::UnimplementedError`.

Note that the returned object will be deallocated by `Xapian` after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the `Xapian` API for use from another language) then you can define a static `operator delete` method in your subclass as shown here: [http://trac.xapian.org/ticket/554#comment:1](http://trac.xapian.org/ticket/554#comment:1)

**Parameters**

- `s` A string containing the serialised parameters.

Reimplemented from `Xapian::Weight`.

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

- xapian/weight.h

### 7.69 Xapian::UnimplementedError Class Reference

`UnimplementedError` indicates an attempt to use an unimplemented feature.
7.69 Xapian::UnimplementedError Class Reference

Inheritance diagram for Xapian::UnimplementedError:

```
Xapian::Error
   `-- Xapian::LogicError
        `-- Xapian::UnimplementedError
```

Public Member Functions

- `UnimplementedError(const std::string &msg_, const std::string &context_=std::string(), int errno_=0)`
  
  General purpose constructor.

- `UnimplementedError(const std::string &msg_, int errno_)`
  
  Construct from message and errno value.

7.69.1 Detailed Description

`UnimplementedError` indicates an attempt to use an unimplemented feature.

7.69.2 Constructor & Destructor Documentation

7.69.2.1 Xapian::UnimplementedError::UnimplementedError(const std::string &msg_, const std::string &context_=std::string(), int errno_=0) [inline, explicit]

General purpose constructor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>msg_</code></td>
<td>Message giving details of the error, intended for human consumption.</td>
</tr>
<tr>
<td><code>context_</code></td>
<td>Optional context information for this error.</td>
</tr>
<tr>
<td><code>errno_</code></td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>
7.69.2.2 Xapian::UnimplementedError::UnimplementedError ( const std::string & msg_, int errno_ ) [inline]

Construct from message and errno value.

Parameters

<table>
<thead>
<tr>
<th>msg_</th>
<th>Message giving details of the error, intended for human consumption.</th>
</tr>
</thead>
<tbody>
<tr>
<td>errno_</td>
<td>Optional errno value associated with this error.</td>
</tr>
</tbody>
</table>

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

- xapian/error.h

7.70 Xapian::Utf8Iterator Class Reference

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

Public Types

- typedef std::input_iterator_tag iterator_category
  
  We implement the semantics of an STL input_iterator.
- typedef unsigned value_type
  
  We implement the semantics of an STL input_iterator.
- typedef size_t difference_type
  
  We implement the semantics of an STL input_iterator.
- typedef const unsigned * pointer
  
  We implement the semantics of an STL input_iterator.
- typedef const unsigned & reference
  
  We implement the semantics of an STL input_iterator.

Public Member Functions

- const char * raw () const
  
  Return the raw const char * pointer for the current position.
- size_t left () const
  
  Return the number of bytes left in the iterator's buffer.
- void assign (const char *p_, size_t len)
  
  Assign a new string to the iterator.
- void assign (const std::string &s)
  
  Assign a new string to the iterator.
- Utf8Iterator (const char *p_)
  
  Create an iterator given a pointer to a null terminated string.
• Utf8Iterator (const char *p_, size_t len)
  Create an iterator given a pointer and a length.
• Utf8Iterator (const std::string &s)
  Create an iterator given a string.
• Utf8Iterator ()
  Create an iterator which is at the end of its iteration.
• unsigned operator* () const
  Get the current Unicode character value pointed to by the iterator.
• Utf8Iterator operator++ (int)
  Move forward to the next Unicode character.
• Utf8Iterator & operator++ ()
  Move forward to the next Unicode character.
• bool operator==(const Utf8Iterator &other) const
  Test two Utf8Iterators for equality.
• bool operator!=(const Utf8Iterator &other) const
  Test two Utf8Iterators for inequality.

7.70.1 Detailed Description
An iterator which returns Unicode character values from a UTF-8 encoded string.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 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. |

7.70.2.2 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. |
7.70.2.3 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. |

7.70.2.4 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.

7.70.3 Member Function Documentation

7.70.3.1 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. |

7.70.3.2 void Xapian::Utf8Iterator::assign ( const std::string & s ) [inline]

Assign a new string to the iterator.
The iterator will forget the string it was iterating through, and return characters from the start of the new string when next called. The string is not copied into the iterator, so it must remain valid while the iteration is in progress.

Parameters

| s    | The string to read. Must not be modified while the iteration is in progress. |

References assign().
Referenced by assign().

7.70.3.3 `size_t Xapian::Utf8Iterator::left ( ) const [inline]`

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

7.70.3.4 `bool Xapian::Utf8Iterator::operator!= ( const Utf8Iterator & other ) const [inline]`

Test two Utf8Iterators for inequality.

Parameters

| other | The Utf8Iterator to compare this one with. |

Returns

true iff the iterators do not point to the same position.

7.70.3.5 `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.

7.70.3.6 `Utf8Iterator Xapian::Utf8Iterator::operator++ ( int ) [inline]`

Move forward to the next Unicode character.

Returns

An iterator pointing to the position before the move.

7.70.3.7 `Utf8Iterator& Xapian::Utf8Iterator::operator++( ) [inline]`

Move forward to the next Unicode character.

Returns

A reference to this object.

7.70.3.8 `bool Xapian::Utf8Iterator::operator== ( const Utf8Iterator & other ) const [inline]`

Test two Utf8Iterators for equality.
220 Class Documentation

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>The Utf8Iterator to compare this one with.</td>
</tr>
</tbody>
</table>

Returns

true iff the iterators point to the same position.

7.70.3.9 `const char* Xapian::Utf8Iterator::raw() const` [inline]

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

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

- `xapian/unicode.h`

7.71 Xapian::ValueCountMatchSpy Class Reference

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

Inheritance diagram for Xapian::ValueCountMatchSpy:

```
Xapian::MatchSpy
    ▲
  Xapian::ValueCountMatchSpy
```

Public Member Functions

- `ValueCountMatchSpy()`
  Construct an empty ValueCountMatchSpy.
- `ValueCountMatchSpy(Xapian::valueno slot_)`
  Construct a MatchSpy which counts the values in a particular slot.
- `size_t get_total() const`
  Return the total number of documents tallied.
- `TermIterator values_begin() const`
Get an iterator over the values seen in the slot.

- TermIterator values_end () const
  End iterator corresponding to values_begin()

- TermIterator top_values_begin (size_t maxvalues) const
  Get an iterator over the most frequent values seen in the slot.

- TermIterator top_values_end (size_t) const
  End iterator corresponding to top_values_begin()

- void operator() (const Xapian::Document &doc, double wt)
  Implementation of virtual operator().

- virtual MatchSpy * clone () const
  Clone the match spy.

- virtual std::string name () const
  Return the name of this match spy.

- virtual std::string serialise () const
  Return this object’s parameters serialised as a single string.

- virtual MatchSpy * unserialise (const std::string &s, const Registry &context) const
  Unserialise parameters.

- virtual std::string serialise_results () const
  Serialise the results of this match spy.

- virtual void merge_results (const std::string &s)
  Unserialise some results, and merge them into this matchspy.

- virtual std::string get_description () const
  Return a string describing this object.

7.71.1 Detailed Description

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

7.71.2 Member Function Documentation

7.71.2.1 virtual MatchSpy * Xapian::ValueCountMatchSpy::clone () const
  [virtual]

Clone the match spy.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be passed information about the results seen by the parent.

If you don’t want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator
delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1
Reimplemented from Xapian::MatchSpy.

7.71.2.2 virtual std::string Xapian::ValueCountMatchSpy::get_description() const
            [virtual]

Return a string describing this object.
This default implementation returns a generic answer, to avoid forcing those deriving
their own MatchSpy subclasses from having to implement this (they may not care what
get_description() gives for their subclass).
Reimplemented from Xapian::MatchSpy.

7.71.2.3 size_t Xapian::ValueCountMatchSpy::get_total() const [inline]

Return the total number of documents tallied.

7.71.2.4 virtual void Xapian::ValueCountMatchSpy::merge_results(const std::string & s) [virtual]

Unserialize some results, and merge them into this matchspy.
The order in which results are merged should not be significant, since this order is not
specified (and will vary depending on the speed of the search in each sub-database).
If you don’t want to support the remote backend in your match spy, you can use the
default implementation which simply throws Xapian::UnimplementedError.

Parameters

| s | A string containing the serialised results. |

Reimplemented from Xapian::MatchSpy.

7.71.2.5 virtual std::string Xapian::ValueCountMatchSpy::name() const
            [virtual]

Return the name of this match spy.
This name is used by the remote backend. It is passed with the serialised parameters
to the remote server so that it knows which class to create.
Return the full namespace-qualified name of your class here - if your class is called
MyApp::FooMatchSpy, return "MyApp::FooMatchSpy" from this method.
If you don’t want to support the remote backend in your match spy, you can use the
default implementation which simply throws Xapian::UnimplementedError.
Reimplemented from Xapian::MatchSpy.
7.71.2.6  void Xapian::ValueCountMatchSpy::operator() ( const Xapian::Document & doc, double wt ) [virtual]

Implementation of virtual operator().
This implementation tallies values for a matching document.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>doc</code></td>
<td>The document to tally values for.</td>
</tr>
<tr>
<td><code>wt</code></td>
<td>The weight of the document (ignored by this class).</td>
</tr>
</tbody>
</table>

Implements Xapian::MatchSpy.

7.71.2.7  virtual std::string Xapian::ValueCountMatchSpy::serialise ( ) const [virtual]

Return this object’s parameters serialised as a single string.
If you don’t want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.
Reimplemented from Xapian::MatchSpy.

7.71.2.8  virtual std::string Xapian::ValueCountMatchSpy::serialise_results ( ) const [virtual]

Serialise the results of this match spy.
If you don’t want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.
Reimplemented from Xapian::MatchSpy.

7.71.2.9  TermIterator Xapian::ValueCountMatchSpy::top_values_begin ( size_t maxvalues ) const

Get an iterator over the most frequent values seen in the slot.
Items will be returned in descending order of frequency. Values with the same frequency will be returned in ascending alphabetical order.
During the iteration, the frequency of the current value can be obtained with the get_termfreq() method on the iterator.

Parameters

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>maxvalues</code></td>
<td>The maximum number of values to return.</td>
</tr>
</tbody>
</table>
7.71.2.10 virtual MatchSpy* Xapian::ValueCountMatchSpy::unserialise ( const std::string & s, const Registry & context ) const [virtual]

Unserialise parameters.

This method unserialises parameters serialised by the serialise() method and allocates and returns a new object initialised with them.

If you don’t want to support the remote backend in your match spy, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with “delete”. - If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

Parameters

| s | A string containing the serialised results. |
| context | Registry object to use for unserialisation to permit MatchSpy sub-classes with sub-MatchSpy objects to be implemented. |

Reimplemented from Xapian::MatchSpy.

7.71.2.11 TermIterator Xapian::ValueCountMatchSpy::values_begin ( ) const

Get an iterator over the values seen in the slot.

Items will be returned in ascending alphabetical order.

During the iteration, the frequency of the current value can be obtained with the get_termfreq() method on the iterator.

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

- xapian/matchspy.h

7.72 Xapian::Valuelterator Class Reference

Class for iterating over document values.

Public Member Functions

- Valuelterator (const Valuelterator &o)
  Copy constructor.
- Valuelterator & operator= (const Valuelterator &o)
  Assignment.
- Valuelterator ()
7.72 Xapian::ValueIterator Class Reference

Default constructor.

- ~ValueIterator ()
  Destructor.
- std::string operator* () const
  Return the value at the current position.
- ValueIterator & operator++ ()
  Advance the iterator to the next position.
- DerefWrapper_ < std::string > operator++ (int)
  Advance the iterator to the next position (postfix version).
- Xapian::docid get_docid () const
  Return the docid at the current position.
- Xapian::valueno get_valueno () const
  Return the value slot number for the current position.
- void skip_to (Xapian::docid docid_or_slot)
  Advance the iterator to document id or value slot docid_or_slot.
- bool check (Xapian::docid docid)
  Check if the specified docid occurs.
- std::string get_description () const
  Return a string describing this object.

7.72.1 Detailed Description

Class for iterating over document values.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 Xapian::ValueIterator::ValueIterator ( ) [inline]

Default constructor.

Creates an uninitialised iterator, which can’t be used before being assigned to, but is sometimes syntactically convenient.

7.72.3 Member Function Documentation

7.72.3.1 bool Xapian::ValueIterator::check ( Xapian::docid docid )

Check if the specified docid occurs.

The caller is required to ensure that the specified document id did actually exists in the database.

This method acts like skip_to() if that can be done at little extra cost, in which case it then returns true. This is how brass and chert databases behave because they store values in streams which allow for an efficient implementation of skip_to().

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
Otherwise it simply checks if a particular docid is present. If it is, it returns true. -
If it isn’t, it returns false, and leaves the position unspecified (and hence the result
of calling methods which depends on the current position, such as get_docid(), are
also unspecified). In this state, next() will advance to the first matching position after
document did, and skip_to() will act as it would if the position was the first matching
position after document did.

Currently the inmemory and remote backends behave in the latter way because they
don’t support streamed values and so skip_to() must check each document it skips over
which is significantly slower.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docid</td>
<td>The document id to check.</td>
</tr>
</tbody>
</table>

7.72.3.2 Xapian::docid Xapian::ValueIterator::get_docid ( ) const

Return the docid at the current position.
If we’re iterating over values of a document, this method will throw Xapian::Invalid-
OperationError.

7.72.3.3 Xapian::valueno Xapian::ValueIterator::get_valueno ( ) const

Return the value slot number for the current position.
If the iterator is over all values in a slot, this returns that slot’s number. If the iterator is
over the values in a particular document, it returns the number of each slot in turn.

7.72.3.4 void Xapian::ValueIterator::skip_to ( Xapian::docid docid_or_slot )

Advance the iterator to document id or value slot docid_or_slot.
If this iterator is over values in a document, then this method advances the iterator to
value slot docid_or_slot, or the first slot after it if there is no value in slot slot.
If this iterator is over values in a particular slot, then this method advances the iterator
to document id docid_or_slot, or the first document id after it if there is no value in the
slot we’re iterating over for document docid_or_slot.

Note: The "two-faced" nature of this method is due to how C++ overloading works. -
Xapian::docid and Xapian::valueno are both typedefs for the same unsigned integer
type, so overloading can’t distinguish them.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>docid_or_slot</td>
<td>The docid/slot to advance to.</td>
</tr>
</tbody>
</table>

The documentation for this class was generated from the following file:
7.73 Xapian::ValueMapPostingSource Class Reference

A posting source which looks up weights in a map using values as the key.

Inheritance diagram for Xapian::ValueMapPostingSource:

```
Xapian::PostingSource
  ↓
Xapian::ValuePostingSource
  ↓
Xapian::ValueMapPostingSource
```

Public Member Functions

- **ValueMapPostingSource (Xapian::valueno slot_)**
  
  Construct a ValueWeightPostingSource.

- **void add_mapping (const std::string &key, double wt)**
  
  Add a mapping.

- **void clear_mappings ()**
  
  Clear all mappings.

- **void set_default_weight (double wt)**
  
  Set a default weight for document values not in the map.

- **double get_weight () const**
  
  Return the weight contribution for the current document.

- **ValueMapPostingSource ∗ clone () const**

  Clone the posting source.

- **std::string name () const**
  
  Name of the posting source class.

- **std::string serialise () const**
**Class Documentation**

Serialise object parameters into a string.

- **ValueMapPostingSource** *unserialise* (const std::string &s) const
  
  Create object given string serialisation returned by *serialise*().

- **void** *init* (const Database &db_)
  
  Set this *PostingSource* to the start of the list of postings.

- **std::string** *get_description* () const
  
  Return a string describing this object.

### 7.73.1 Detailed Description

A posting source which looks up weights in a map using values as the key.

This allows will return entries for all documents in the given database which have a value in the slot specified. The values will be mapped to the corresponding weight in the weight map. If there is no mapping for a particular value, the default weight will be returned (which itself defaults to 0.0).

### 7.73.2 Constructor & Destructor Documentation

#### 7.73.2.1 Xapian::ValueMapPostingSource::ValueMapPostingSource

Construct a *ValueWeightPostingSource*.

**Parameters**

- **slot_** | The value slot to read values from.

### 7.73.3 Member Function Documentation

#### 7.73.3.1 void Xapian::ValueMapPostingSource::add_mapping

Add a mapping.

**Parameters**

- **key** | The key looked up from the value slot.
- **wt** | The weight to give this key.

#### 7.73.3.2 void Xapian::ValueMapPostingSource::clear_mappings

Clear all mappings.
Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call \texttt{init()} on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the \texttt{PostingSource} may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by \texttt{Xapian} after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the \texttt{Xapian} API for use from another language) then you can define a static \texttt{operator delete} method in your subclass as shown here: \url{http://trac.xapian.org/ticket/554#comment:1}

Reimplemented from \texttt{Xapian::PostingSource}.

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own \texttt{PostingSource} subclass from having to implement this (they may not care what \texttt{get_description()} gives for their subclass).

Reimplemented from \texttt{Xapian::PostingSource}.

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" \texttt{PostingSource} subclasses.

This method may assume that it will only be called when there is a "current document". In detail: \texttt{Xapian} will always call \texttt{init()} on a \texttt{PostingSource} before calling this for the first time. It will also only call this if the \texttt{PostingSource} reports that it is pointing to a valid document (ie, it will not call it before calling at least one of \texttt{next()}, \texttt{skip_to()} or \texttt{check()}, and will ensure that the \texttt{PostingSource} is not at the end by calling \texttt{at_end()}).

Reimplemented from \texttt{Xapian::PostingSource}. 

---

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
void Xapian::ValueMapPostingSource::init ( const Database & db )
   [virtual]

Set this PostingSource to the start of the list of postings.
This is called automatically by the matcher prior to each query being processed.
If a PostingSource is used for multiple searches, init() will therefore be called multiple times, and must handle this by using the database passed in the most recent call.

Parameters

| db | The database which the PostingSource should iterate through. |

Note: the database supplied to this method must not be modified: in particular, the reopen() method should not be called on it.
Note: in the case of a multi-database search, a separate PostingSource will be used for each database (the separate PostingSources will be obtained using clone()), and each PostingSource will be passed one of the sub-databases as the db parameter here. The db parameter will therefore always refer to a single database. All docids passed to, or returned from, the PostingSource refer to docids in that single database, rather than in the multi-database.

Reimplemented from Xapian::ValuePostingSource.

std::string Xapian::ValueMapPostingSource::name ( ) const  [virtual]

Name of the posting source class.
This is used when serialising and unserialising posting sources; for example, for performing remote searches.
If the subclass is in a C++ namespace, the namespace should be included in the name, using "::" as a separator. For example, for a PostingSource subclass called "-FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".
This should only be implemented if serialise() and unserialise() are also implemented. The default implementation returns an empty string.
If this returns an empty string, Xapian will assume that serialise() and unserialise() are not implemented.
Reimplemented from Xapian::PostingSource.

std::string Xapian::ValueMapPostingSource::serialise ( ) const  [virtual]

Serialise object parameters into a string.
The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.
If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError. Reimplemented from Xapian::PostingSource.

### 7.73.3.9 void Xapian::ValueMapPostingSource::set_default_weight ( double wt )

Set a default weight for document values not in the map.

**Parameters**

- **wt** | The weight to set as the default.

### 7.73.3.10 ValueMapPostingSource Xapian::ValueMapPostingSource::unserialise ( const std::string & s ) const [virtual]

Create object given string serialisation returned by serialise(). Note that the returned object will be deallocated by Xapian after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here:  
http://trac.xapian.-org/ticket/554#comment:1

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

**Parameters**

- **s** | A serialised instance of this PostingSource subclass.

Reimplemented from Xapian::PostingSource.

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

- xapian/postingsource.h

### 7.74 Xapian::ValuePostingSource Class Reference

A posting source which generates weights from a value slot.
Inheritance diagram for Xapian::ValuePostingSource:

Public Member Functions

- `ValuePostingSource (Xapian::valueno slot_)`
  Construct a `ValuePostingSource`.
- `Xapian::doccount get_termfreq_min () const`
  A lower bound on the number of documents this object can return.
- `Xapian::doccount get_termfreq_est () const`
  An estimate of the number of documents this object can return.
- `Xapian::doccount get_termfreq_max () const`
  An upper bound on the number of documents this object can return.
- `void next (double min_wt)`
  Advance the current position to the next matching document.
- `void skip_to (Xapian::docid min_docid, double min_wt)`
  Advance to the specified docid.
- `bool check (Xapian::docid min_docid, double min_wt)`
  Check if the specified docid occurs.
- `bool at_end () const`
  Return true if the current position is past the last entry in this list.
- `Xapian::docid get_docid () const`
  Return the current docid.
- `void init (const Database &db_)`
  Set this PostingSource to the start of the list of postings.

Protected Attributes

- `Xapian::Database db`
  The database we’re reading values from.
- `Xapian::valueno slot`
  The slot we’re reading values from.
- `Xapian::ValueIterator value_it`
  Value stream iterator.
- `bool started`
  Flag indicating if we’ve started (true if we have).
- `Xapian::doccount termfreq_min`
A lower bound on the term frequency.

- Xapian::doccount termfreq_est
  An estimate of the term frequency.
- Xapian::doccount termfreq_max
  An upper bound on the term frequency.

7.74.1 Detailed Description

A posting source which generates weights from a value slot.

This is a base class for classes which generate weights using values stored in the specified slot. For example, ValueWeightPostingSource uses sortable_unserialise to convert values directly to weights.

The upper bound on the weight returned is set to DBL_MAX. Subclasses should call set_maxweight() in their init() methods after calling ValuePostingSource::init() if they know a tighter bound on the weight.

7.74.2 Constructor & Destructor Documentation

7.74.2.1 Xapian::ValuePostingSource::ValuePostingSource

Construct a ValuePostingSource.

Parameters

- slot_ The value slot to read values from.

7.74.3 Member Function Documentation

7.74.3.1 bool Xapian::ValuePostingSource::at_end() const [virtual]

Return true if the current position is past the last entry in this list.

At least one of next(), skip_to() or check() will be called before this method is first called.

Implements Xapian::PostingSource.

7.74.3.2 bool Xapian::ValuePostingSource::check( Xapian::docid did, double min_wt ) [virtual]

Check if the specified docid occurs.

The caller is required to ensure that the specified document id did actually exists in the database. If it does, it must move to that document id, and return true. If it does not, it may either:
• return true, having moved to a definite position (including "at_end"), which must be the same position as skip_to() would have moved to.

or

• return false, having moved to an "indeterminate" position, such that a subsequent call to next() or skip_to() will move to the next matching position after did.

Generally, this method should act like skip_to() and return true if that can be done at little extra cost. Otherwise it should simply check if a particular docid is present, returning true if it is, and false if it isn’t.

The default implementation calls skip_to() and always returns true.

Xapian will always call init() on a PostingSource before calling this for the first time.

Note: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the init() method for details.

Parameters

<table>
<thead>
<tr>
<th>did</th>
<th>The document id to check.</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_wt</td>
<td>The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).</td>
</tr>
</tbody>
</table>

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource, and Xapian::Lat-LongDistancePostingSource.

7.74.3.3 Xapian::docid Xapian::ValuePostingSource::get_docid ( ) const [virtual]

Return the current docid.

This method may assume that it will only be called when there is a "current document". See get_weight() for details.

Note: in the case of a multi-database search, the returned docid should be in the single subdatabase relevant to this posting source. See the init() method for details.

Implements Xapian::PostingSource.

7.74.3.4 Xapian::doccount Xapian::ValuePostingSource::get_termfreq_est ( ) const [virtual]

An estimate of the number of documents this object can return.

It must always be true that:

get_termfreq_min() <= get_termfreq_est() <= get_termfreq_max()
Xapian will always call init() on a PostingSource before calling this for the first time.
Implements Xapian::PostingSource.

7.74.3.5  Xapian::doccount Xapian::ValuePostingSource::get_termfreq_max ( )
cost [virtual]

An upper bound on the number of documents this object can return.
Xapian will always call init() on a PostingSource before calling this for the first time.
Implements Xapian::PostingSource.

7.74.3.6  Xapian::doccount Xapian::ValuePostingSource::get_termfreq_min ( )
cost [virtual]

A lower bound on the number of documents this object can return.
Xapian will always call init() on a PostingSource before calling this for the first time.
Implements Xapian::PostingSource.

7.74.3.7  void Xapian::ValuePostingSource::init ( const Database & db )
[virtual]

Set this PostingSource to the start of the list of postings.
This is called automatically by the matcher prior to each query being processed.
If a PostingSource is used for multiple searches, init() will therefore be called multiple
times, and must handle this by using the database passed in the most recent call.

Parameters

| db | The database which the PostingSource should iterate through. |

Note: the database supplied to this method must not be modified: in particular, the
reopen() method should not be called on it.
Note: in the case of a multi-database search, a separate PostingSource will be used for
each database (the separate PostingSources will be obtained using clone()), and each
PostingSource will be passed one of the sub-databases as the db parameter here. The
db parameter will therefore always refer to a single database. All docids passed to, or
returned from, the PostingSource refer to docids in that single database, rather than in
the multi-database.
Implements Xapian::PostingSource.
Reimplemented in Xapian::ValueMapPostingSource, Xapian::DecreasingValueWeight-
PostingSource, Xapian::LatLongDistancePostingSource, and Xapian::ValueWeight-
PostingSource.
7.74.3.8 void Xapian::ValuePostingSource::next ( double \textit{min\_wt} ) [virtual]

Advance the current position to the next matching document.

The \textit{PostingSource} starts before the first entry in the list, so \textit{next()} must be called before any methods which need the context of the current position.

\textbf{Xapian} will always call \textit{init()} on a \textit{PostingSource} before calling this for the first time.

\textbf{Parameters}

\begin{itemize}
\item \textit{min\_wt} The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).
\end{itemize}

\textbf{Implements} \textit{Xapian::PostingSource}.

\textbf{Reimplemented} in \textit{Xapian::DecreasingValueWeightPostingSource}, and \textit{Xapian::Lat-LongDistancePostingSource}.

7.74.3.9 void Xapian::ValuePostingSource::skip_to ( Xapian::docid \textit{did}, double \textit{min\_wt} ) [virtual]

Advance to the specified docid.

If the specified docid isn’t in the list, position ourselves on the first document after it (or \textit{at\_end()} if no greater docids are present).

If the current position is already the specified docid, this method will leave the position unmodified.

If the specified docid is earlier than the current position, the behaviour is unspecified. A sensible behaviour would be to leave the current position unmodified, but it is also reasonable to move to the specified docid.

The default implementation calls \textit{next()} repeatedly, which works but \textit{skip\_to()} can often be implemented much more efficiently.

\textbf{Xapian} will always call \textit{init()} on a \textit{PostingSource} before calling this for the first time.

\textbf{Note}: in the case of a multi-database search, the docid specified is the docid in the single subdatabase relevant to this posting source. See the \textit{init()} method for details.

\textbf{Parameters}

\begin{itemize}
\item \textit{did} The document id to advance to.
\item \textit{min\_wt} The minimum weight contribution that is needed (this is just a hint which subclasses may ignore).
\end{itemize}

\textbf{Reimplemented} from \textit{Xapian::PostingSource}.

\textbf{Reimplemented} in \textit{Xapian::DecreasingValueWeightPostingSource}, and \textit{Xapian::Lat-LongDistancePostingSource}. 

---

Generated on Fri May 3 2013 06:18:28 for xapian-core by Doxygen
7.74.4 Member Data Documentation

7.74.4.1 Xapian::doccount Xapian::ValuePostingSource::termfreq_est
[protected]

An estimate of the term frequency.
Subclasses should set this if they are overriding the next(), skip_to() or check() methods.

7.74.4.2 Xapian::doccount Xapian::ValuePostingSource::termfreq_max
[protected]

An upper bound on the term frequency.
Subclasses should set this if they are overriding the next(), skip_to() or check() methods.

7.74.4.3 Xapian::doccount Xapian::ValuePostingSource::termfreq_min
[protected]

A lower bound on the term frequency.
Subclasses should set this if they are overriding the next(), skip_to() or check() methods to return fewer documents.

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

- xapian/postingsource.h

7.75 Xapian::ValueRangeProcessor Struct Reference

Base class for value range processors.
Inheritance diagram for Xapian::ValueRangeProcessor:

```
Xapian::ValueRangeProcessor
     /       \
  Xapian::StringValueRangeProcessor
  /                      /     \
Xapian::DateValueRangeProcessor Xapian::NumberValueRangeProcessor
```

Public Member Functions

- virtual ~ValueRangeProcessor ()
  Destructor.
- virtual Xapian::valueno operator() (std::string &begin, std::string &end)=0
  Check for a valid range of this type.

7.75.1 Detailed Description

Base class for value range processors.

7.75.2 Member Function Documentation

7.75.2.1 virtual Xapian::valueno Xapian::ValueRangeProcessor::operator() (std::string &begin, std::string &end) [pure virtual]

Check for a valid range of this type.

Parameters

<table>
<thead>
<tr>
<th>in,out</th>
<th>begin</th>
<th>The start of the range as specified in the query string by the user. This parameter is a non-const reference so the ValueRangeProcessor can modify it to return the value to start the range with.</th>
</tr>
</thead>
<tbody>
<tr>
<td>in,out</td>
<td>end</td>
<td>The end of the range. This is also a non-const reference so it can be modified.</td>
</tr>
</tbody>
</table>
Returns

If this \texttt{ValueRangeProcessor} recognises the range BEGIN..END it returns the value slot number to range filter on. Otherwise it returns \texttt{Xapian::BAD\_VALUENO}.

Implemented in \texttt{Xapian::NumberValueRangeProcessor}, \texttt{Xapian::DateValueRangeProcessor}, and \texttt{Xapian::StringValueRangeProcessor}.

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

\begin{itemize}
\item xapian/queryparser.h
\end{itemize}

\section*{Public Member Functions}

\begin{itemize}
\item \texttt{ValueSetMatchDecider (Xapian::valueno slot, bool inclusive_)}
\end{itemize}

Construct a \texttt{ValueSetMatchDecider}.

\begin{itemize}
\item \texttt{void add\_value (const std::string &value)}
\end{itemize}

Add a value to the test set.

\begin{itemize}
\item \texttt{void remove\_value (const std::string &value)}
\end{itemize}

Remove a value from the test set.

\begin{itemize}
\item \texttt{bool operator() (const Xapian::Document &doc) const}
\end{itemize}

Decide whether we want a particular document to be in the \texttt{MSet}.
7.76.1 Detailed Description

*MatchDecider* filtering results based on whether document values are in a user-defined set.

### 7.76.2 Constructor & Destructor Documentation

#### 7.76.2.1

```
Xapian::ValueSetMatchDecider::ValueSetMatchDecider ( 
Xapian::valueno slot, bool inclusive_ ) [inline]
```

Construct a *ValueSetMatchDecider*.

#### Parameters

<table>
<thead>
<tr>
<th>slot</th>
<th>The value slot number to look in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>inclusive_</td>
<td>If true, match decider accepts documents which have a value in the specified slot which is a member of the test set; if false, match decider accepts documents which do not have a value in the specified slot.</td>
</tr>
</tbody>
</table>

#### 7.76.3 Member Function Documentation

#### 7.76.3.1

```
void Xapian::ValueSetMatchDecider::add_value ( const std::string & value ) [inline]
```

Add a value to the test set.

#### Parameters

| value | The value to add to the test set. |

#### 7.76.3.2

```
bool Xapian::ValueSetMatchDecider::operator() ( const Xapian::Document & doc ) const [virtual]
```

Decide whether we want a particular document to be in the *MSet*.

#### Parameters

| doc | The document to test. |

#### Returns

true if the document is acceptable, or false if the document should be excluded from the *MSet*.

Implements *Xapian::MatchDecider*.
7.76.3.3 void Xapian::ValueSetMatchDecider::remove_value (const std::string & value) [inline]

Remove a value from the test set.

Parameters

| value | The value to remove from the test set. |

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

- xapian/valuesetmatchdecider.h

7.77 Xapian::ValueWeightPostingSource Class Reference

A posting source which reads weights from a value slot.

Inheritance diagram for Xapian::ValueWeightPostingSource:

```
Xapian::PostingSource

Xapian::ValuePostingSource

Xapian::ValueWeightPostingSource

Xapian::DecreasingValueWeightPostingSource
```

Public Member Functions

- **ValueWeightPostingSource (Xapian::valueno slot_)**
  Construct a **ValueWeightPostingSource**.
• double get_weight () const
  Return the weight contribution for the current document.
• ValueWeightPostingSource * clone () const
  Clone the posting source.
• std::string name () const
  Name of the posting source class.
• std::string serialise () const
  Serialise object parameters into a string.
• ValueWeightPostingSource * unserialise (const std::string &s) const
  Create object given string serialisation returned by serialise().
• void init (const Database &db_)
  Set this PostingSource to the start of the list of postings.
• std::string get_description () const
  Return a string describing this object.

7.77.1 Detailed Description

A posting source which reads weights from a value slot.

This returns entries for all documents in the given database which have a non empty values in the specified slot. It returns a weight calculated by applying sortable_unserialise to the value stored in the slot (so the values stored should probably have been calculated by applying sortable_serialise to a floating point number at index time).

The upper bound on the weight returned is set using the upper bound on the values in the specified slot, or DBL_MAX if value bounds aren’t supported by the current backend.

For efficiency, this posting source doesn’t check that the stored values are valid in any way, so it will never raise an exception due to invalid stored values. In particular, it doesn’t ensure that the unserialised values are positive, which is a requirement for weights. The behaviour if the slot contains values which unserialise to negative values is undefined.

7.77.2 Constructor & Destructor Documentation

7.77.2.1 Xapian::ValueWeightPostingSource::ValueWeightPostingSource ( Xapian::valueno slot_ )

Construct a ValueWeightPostingSource.

Parameters

slot_ The value slot to read values from.

7.77.3 Member Function Documentation
7.77.3.1 ValueWeightPostingSource::clone

Clone the posting source.

The clone should inherit the configuration of the parent, but need not inherit the state. ie, the clone does not need to be in the same iteration position as the original: the matcher will always call init() on the clone before attempting to move the iterator, or read the information about the current position of the iterator.

This may return NULL to indicate that cloning is not supported. In this case, the PostingSource may only be used with a single-database search.

The default implementation returns NULL.

Note that the returned object will be deallocated by Xapian after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

Reimplemented from Xapian::PostingSource.
Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.77.3.2 std::string ValueWeightPostingSource::get_description()

Return a string describing this object.

This default implementation returns a generic answer. This default it provided to avoid forcing those deriving their own PostingSource subclass from having to implement this (they may not care what get_description() gives for their subclass).

Reimplemented from Xapian::PostingSource.
Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.77.3.3 double ValueWeightPostingSource::get_weight()

Return the weight contribution for the current document.

This default implementation always returns 0, for convenience when implementing "weight-less" PostingSource subclasses.

This method may assume that it will only be called when there is a "current document". In detail: Xapian will always call init() on a PostingSource before calling this for the first time. It will also only call this if the PostingSource reports that it is pointing to a valid document (ie, it will not call it before calling at least one of next(), skip_to() or check(), and will ensure that the PostingSource is not at the end by calling at_end()).

Reimplemented from Xapian::PostingSource.
Reimplemented in Xapian::DecreasingValueWeightPostingSource.
7.7.7.3.4 void Xapian::ValueWeightPostingSource::init ( const Database & db )
     [virtual]

Set this PostingSource to the start of the list of postings.
This is called automatically by the matcher prior to each query being processed.
If a PostingSource is used for multiple searches, init() will therefore be called multiple
 times, and must handle this by using the database passed in the most recent call.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db</td>
<td>The database which the PostingSource should iterate through.</td>
</tr>
</tbody>
</table>

Note: the database supplied to this method must not be modified: in particular, the
reopen() method should not be called on it.
Note: in the case of a multi-database search, a separate PostingSource will be used for
each database (the separate PostingSources will be obtained using clone()), and each
PostingSource will be passed one of the sub-databases as the db parameter here. The
db parameter will therefore always refer to a single database. All docids passed to, or
returned from, the PostingSource refer to docids in that single database, rather than in
the multi-database.
Reimplemented from Xapian::ValuePostingSource.
Reimplemented in Xapian::DecreasingValueWeightPostingSource.

7.7.7.5 std::string Xapian::ValueWeightPostingSource::name ( ) const
     [virtual]

Name of the posting source class.
This is used when serialising and unserialising posting sources; for example, for per-
forming remote searches.
If the subclass is in a C++ namespace, the namespace should be included in the
name, using "::" as a separator. For example, for a PostingSource subclass called "-FooPostingSource" in the "Xapian" namespace the result of this call should be "Xapian::FooPostingSource".
This should only be implemented if serialise() and unserialise() are also implemented.
The default implementation returns an empty string.
If this returns an empty string, Xapian will assume that serialise() and unserialise() are
not implemented.
Reimplemented from Xapian::PostingSource.
Reimplemented in Xapian::DecreasingValueWeightPostingSource.
Serialise object parameters into a string.

The serialised parameters should represent the configuration of the posting source, but need not (indeed, should not) represent the current iteration state.

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented from Xapian::PostingSource.

Reimplemented in Xapian::DecreasingValueWeightPostingSource.

---

Create object given string serialisation returned by serialise().

Note that the returned object will be deallocated by Xapian after use with "delete". If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Parameters

- `s` A serialised instance of this PostingSource subclass.
Inheritance diagram for Xapian::Weight:

```
Xapian::Weight
  \|-- Xapian::BM25Weight
  \-- Xapian::BoolWeight
  \-- Xapian::TfIdfWeight
  \-- Xapian::TfidfWeight
```

Public Member Functions

- virtual ~Weight ()
  Virtual destructor, because we have virtual methods.
- virtual Weight * clone () const =0
  Clone this object.
- virtual std::string name () const
  Return the name of this weighting scheme.
- virtual std::string serialise () const
  Return this object's parameters serialised as a single string.
- virtual Weight * unserialise (const std::string &s) const
  Unserialise parameters.
- virtual double get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const =0
  Calculate the weight contribution for this object's term to a document.
- virtual double get_maxpart () const =0
  Return an upper bound on what get_sumpart() can return for any document.
- virtual double get_sumextra (Xapian::termcount doclen) const =0
  Calculate the term-independent weight component for a document.
- virtual double get_maxextra () const =0
  Return an upper bound on what get_sumextra() can return for any document.

Protected Types

- enum stat_flags
  Stats which the weighting scheme can use (see need_stat()).

Protected Member Functions

- void need_stat (stat_flags flag)
  Tell Xapian that your subclass will want a particular statistic.
7.78 Xapian::Weight Class Reference

- virtual void init (double factor)=0
  
  Allow the subclass to perform any initialisation it needs to.
- Weight (const Weight &)
  
  Don't allow copying.
- Weight ()
  
  Default constructor, needed by subclass constructors.
- Xapian::doccount get_collection_size () const
  
  The number of documents in the collection.
- Xapian::doccount get_rset_size () const
  
  The number of documents marked as relevant.
- Xapian::doclength get_average_length () const
  
  The average length of a document in the collection.
- Xapian::doccount get_termfreq () const
  
  The number of documents which this term indexes.
- Xapian::doccount get_reltermfreq () const
  
  The number of relevant documents which this term indexes.
- Xapian::termcount get_query_length () const
  
  The length of the query.
- Xapian::termcount get_wqf () const
  
  The within-query-frequency of this term.
- Xapian::termcount get_doclength_upper_bound () const
  
  An upper bound on the maximum length of any document in the database.
- Xapian::termcount get_doclength_lower_bound () const
  
  A lower bound on the minimum length of any document in the database.
- Xapian::termcount get_wdf_upper_bound () const
  
  An upper bound on the wdf of this term.

7.78.1 Detailed Description

Abstract base class for weighting schemes.

7.78.2 Constructor & Destructor Documentation

7.78.2.1 virtual Xapian::Weight::~Weight ( ) [virtual]

Virtual destructor, because we have virtual methods.

7.78.2.2 Xapian::Weight::Weight ( const Weight & ) [protected]

Don't allow copying.

This would ideally be private, but that causes a compilation error with GCC 4.1 (which appears to be a bug).
7.78.3 Member Function Documentation

7.78.3.1 virtual Weight* Xapian::Weight::clone() const [pure virtual]

Clone this object.
This method allocates and returns a copy of the object it is called on.
If your subclass is called FooWeight and has parameters a and b, then you would implement FooWeight::clone() like so:

```
FooWeight* FooWeight::clone() const { return new FooWeight(a, b); }
```

Note that the returned object will be deallocated by Xapian after use with "delete". -
If you want to handle the deletion in a special way (for example when wrapping the
Xapian API for use from another language) then you can define a static operator
delete method in your subclass as shown here: http://trac.xapian.-
org/ticket/554#comment:1

7.78.3.2 Xapian::termcount Xapian::Weight::get_doclength_lower_bound() const [inline, protected]

A lower bound on the minimum length of any document in the database.
This bound does not include any zero-length documents.
This should only be used by get_maxpart() and get_maxextra().

7.78.3.3 Xapian::termcount Xapian::Weight::get_doclength_upper_bound() const [inline, protected]

An upper bound on the maximum length of any document in the database.
This should only be used by get_maxpart() and get_maxextra().

7.78.3.4 virtual double Xapian::Weight::get_maxextra() const [pure virtual]

Return an upper bound on what get_sumextra() can return for any document.
This information is used by the matcher to perform various optimisations, so strive to
make the bound as tight as possible.
Implemented in Xapian::TradWeight, Xapian::BM25Weight, Xapian::TfIdfWeight, and -
Xapian::BoolWeight.

7.78.3.5 virtual double Xapian::Weight::get_maxpart() const [pure virtual]

Return an upper bound on what get_sumpart() can return for any document.
This information is used by the matcher to perform various optimisations, so strive to
make the bound as tight as possible.
Implemented in Xapian::TradWeight, Xapian::BM25Weight, Xapian::TfIdfWeight, and Xapian::BoolWeight.

7.78.3.6 virtual double Xapian::Weight::get_sumextra (Xapian::termcount doclen) const [pure virtual]

Calculate the term-independent weight component for a document.
The parameter gives information about the document which may be used in the calculations:

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>doclen</strong></td>
</tr>
</tbody>
</table>

Implemented in Xapian::TradWeight, Xapian::BM25Weight, Xapian::TfIdfWeight, and Xapian::BoolWeight.

7.78.3.7 virtual double Xapian::Weight::get_sumpart (Xapian::termcount wdf, Xapian::termcount doclen) const [pure virtual]

Calculate the weight contribution for this object’s term to a document.
The parameters give information about the document which may be used in the calculations:

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>wdf</strong></td>
</tr>
<tr>
<td><strong>doclen</strong></td>
</tr>
</tbody>
</table>

Implemented in Xapian::TradWeight, Xapian::BM25Weight, Xapian::TfIdfWeight, and Xapian::BoolWeight.

7.78.3.8 Xapian::termcount Xapian::Weight::get_wdf_upper_bound () const [inline, protected]

An upper bound on the wdf of this term.
This should only be used by get_maxpart() and get_maxextra().

7.78.3.9 virtual void Xapian::Weight::init (double factor) [protected, pure virtual]

Allow the subclass to perform any initialisation it needs to.
Parameters

| factor | Any scaling factor (e.g. from OP_SCALE_WEIGHT). If the Weight object is for the term-independent weight supplied by get_sumextra()/get_maxextra(), then init(0.0) is called (starting from Xapian 1.2.11 and 1.3.1 - earlier versions failed to call init() for such Weight objects). |

7.78.3.10 virtual std::string Xapian::Weight::name() const [virtual]

Return the name of this weighting scheme.

This name is used by the remote backend. It is passed along with the serialised parameters to the remote server so that it knows which class to create.

Return the full namespace-qualified name of your class here - if your class is called Foo-Weight, return "FooWeight" from this method (Xapian::BM25Weight returns "Xapian::BM25Weight" here).

If you don't want to support the remote backend, you can use the default implementation which simply returns an empty string.

Reimplemented in Xapian::TradWeight, Xapian::BM25Weight, Xapian::TfIdfWeight, and Xapian::BoolWeight.

7.78.3.11 void Xapian::Weight::need_stat(stat_flags flag) [inline, protected]

Tell Xapian that your subclass will want a particular statistic.

Some of the statistics can be costly to fetch or calculate, so Xapian needs to know which are actually going to be used. You should call need_stat() from your constructor for each such statistic.

Parameters

| flag | The stat_flags value for a required statistic. |

7.78.3.12 virtual std::string Xapian::Weight::serialise() const [virtual]

Return this object's parameters serialised as a single string.

If you don't want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Reimplemented in Xapian::TradWeight, Xapian::BM25Weight, Xapian::TfIdfWeight, and Xapian::BoolWeight.
virtual Weight * Xapian::Weight::unserialise ( const std::string & s ) const  
[virtual]

Unserialise parameters.

This method unserialises parameters serialised by the serialise() method and allocates and returns a new object initialised with them.

If you don’t want to support the remote backend, you can use the default implementation which simply throws Xapian::UnimplementedError.

Note that the returned object will be deallocated by Xapian after use with "delete". - If you want to handle the deletion in a special way (for example when wrapping the Xapian API for use from another language) then you can define a static operator delete method in your subclass as shown here: http://trac.xapian.org/ticket/554#comment:1

Parameters

| s | A string containing the serialised parameters. |

Reimplemented in Xapian::TradWeight, Xapian::BM25Weight, Xapian::TfIdfWeight, and Xapian::BoolWeight.

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

- `xapian/weight.h`

### 7.79 Xapian::WritableDatabase Class Reference

This class provides read/write access to a database.

Inheritance diagram for Xapian::WritableDatabase:
Public Member Functions

- virtual ∼WritableDatabase ()
  
  Destroy this handle on the database.
- WritableDatabase ()
  
  Create an empty WritableDatabase.
- WritableDatabase (const std::string &path, int action)
  
  Open a database for update, automatically determining the database backend to use.
- WritableDatabase (const WritableDatabase &other)
  
  Copying is allowed.
- void operator= (const WritableDatabase &other)
  
  Assignment is allowed.
- void commit ()
  
  Commit any pending modifications made to the database.
- void flush ()
  
  Pre-1.1.0 name for commit().
- void begin_transaction (bool flushed=true)
  
  Begin a transaction.
- void commit_transaction ()
  
  Complete the transaction currently in progress.
- void cancel_transaction ()
  
  Abort the transaction currently in progress, discarding the pending modifications made to the database.
- Xapian::docid add_document (const Xapian::Document &document)
  
  Add a new document to the database.
- void delete_document (Xapian::docid did)
  
  Delete a document from the database.
- void delete_document (const std::string &unique_term)
  
  Delete any documents indexed by a term from the database.
- void replace_document (Xapian::docid did, const Xapian::Document &document)
  
  Replace a given document in the database.
- Xapian::docid replace_document (const std::string &unique_term, const Xapian::Document &document)
  
  Replace any documents matching a term.
- void add_spelling (const std::string &word, Xapian::termcount freqinc=1) const
  
  Add a word to the spelling dictionary.
- void remove_spelling (const std::string &word, Xapian::termcount freqdec=1) const
  
  Remove a word from the spelling dictionary.
- void add_synonym (const std::string &term, const std::string &synonym) const
  
  Add a synonym for a term.
- void remove_synonym (const std::string &term, const std::string &synonym) const
Remove a synonym for a term.
• void clear_synonyms (const std::string &term) const

Remove all synonyms for a term.
• void set_metadata (const std::string &key, const std::string &value)
  Set the user-specified metadata associated with a given key.
• std::string get_description () const
  Return a string describing this object.

### 7.79.1 Detailed Description

This class provides read/write access to a database.

### 7.79.2 Constructor & Destructor Documentation

#### 7.79.2.1 virtual Xapian::WritableDatabase::~WritableDatabase () [virtual]

Destroy this handle on the database.
If no other handles to this database remain, the database will be closed.
If a transaction is active cancel_transaction() will be implicitly called; if no transaction is active commit() will be implicitly called, but any exception will be swallowed (because throwing exceptions in C++ destructors is problematic). If you aren’t using transactions and want to know about any failure to commit changes, call commit() explicitly before the destructor gets called.

#### 7.79.2.2 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

<table>
<thead>
<tr>
<th>path</th>
<th>directory that the database is stored in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>one of:</td>
</tr>
<tr>
<td>• Xapian::DB_CREATE_OR_OPEN open for read/write; create if no db exists</td>
<td></td>
</tr>
<tr>
<td>• Xapian::DB_CREATE create new database; fail if db exists</td>
<td></td>
</tr>
<tr>
<td>• Xapian::DB_CREATE_OR_OVERWRITE overwrite existing db; create if none exists</td>
<td></td>
</tr>
<tr>
<td>• Xapian::DB_OPEN open for read/write; fail if no db exists</td>
<td></td>
</tr>
</tbody>
</table>
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Xapian::Database::CorruptError</code></td>
<td>will be thrown if the database is in a corrupt state.</td>
</tr>
<tr>
<td><code>Xapian::Database::LockError</code></td>
<td>will be thrown if a lock couldn't be acquired on the database.</td>
</tr>
</tbody>
</table>

7.79.2.3 `Xapian::WritableDatabase::WritableDatabase ( const WritableDatabase & other )`

Copying is allowed.
The internals are reference counted, so copying is cheap.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>other</code></td>
<td>The object to copy.</td>
</tr>
</tbody>
</table>

7.79.3 Member Function Documentation

7.79.3.1 `Xapian::docid Xapian::WritableDatabase::add_document ( const Xapian::Document & document )`

Add a new document to the database.

This method adds the specified document to the database, returning a newly allocated document ID. Automatically allocated document IDs come from a per-database monotonically increasing counter, so IDs from deleted documents won't be reused.

If you want to specify the document ID to be used, you should call `replace_document()` instead.

Note that changes to the database won't be immediately committed to disk; see `commit()` 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 `commit()` is called or the database is closed).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>document</code></td>
<td>The new document to be added.</td>
</tr>
</tbody>
</table>

Returns

The document ID of the newly added document.

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Xapian::Database::Error</code></td>
<td>will be thrown if a problem occurs while writing to the database.</td>
</tr>
</tbody>
</table>
Xapian::WritableDatabase Class Reference

Xapian::Database::CorruptError will be thrown if the database is in a corrupt state.

7.79.3.2 void Xapian::WritableDatabase::add_spelling ( const std::string & word, Xapian::termcount freqinc = 1 ) const

Add a word to the spelling dictionary.
If the word is already present, its frequency is increased.

Parameters

<table>
<thead>
<tr>
<th>word</th>
<th>The word to add.</th>
</tr>
</thead>
<tbody>
<tr>
<td>freqinc</td>
<td>How much to increase its frequency by (default 1).</td>
</tr>
</tbody>
</table>

7.79.3.3 void Xapian::WritableDatabase::add_synonym ( const std::string & term, const std::string & synonym ) const

Add a synonym for a term.

Parameters

<table>
<thead>
<tr>
<th>term</th>
<th>The term to add a synonym for.</th>
</tr>
</thead>
<tbody>
<tr>
<td>synonym</td>
<td>The synonym to add. If this is already a synonym for term, then no action is taken.</td>
</tr>
</tbody>
</table>

7.79.3.4 void Xapian::WritableDatabase::begin_transaction ( bool flushed = true )

Begin a transaction.

In Xapian a transaction is a group of modifications to the database which are linked such that either all will be applied simultaneously or none will be applied at all. Even in the case of a power failure, this characteristic should be preserved (as long as the filesystem isn’t corrupted, etc).

A transaction is started with begin_transaction() and can either be committed by calling commit_transaction() or aborted by calling cancel_transaction().

By default, a transaction implicitly calls commit() before and after so that the modifications stand and fall without affecting modifications before or after.

The downside of these implicit calls to commit() is that small transactions can harm indexing performance in the same way that explicitly calling commit() 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 commit() 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.

Parameters

| flushed | Is this a flushed transaction? By default transactions are "flushed", which means that committing a transaction will ensure those changes are permanently written to the database. By contrast, unflushed transactions only ensure that changes within the transaction are either all applied or all aren’t. |
|---------|

Exceptions

<table>
<thead>
<tr>
<th>Xapian::UnimplementedError</th>
<th>will be thrown if transactions are not available for this database type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::InvalidOperationError</td>
<td>will be thrown if this is called at an invalid time, such as when a transaction is already in progress.</td>
</tr>
</tbody>
</table>

7.79.3.5 void Xapian::WritableDatabase::cancel_transaction ( )

Abort the transaction currently in progress, discarding the pending modifications made to the database.

If an error occurs in this method, an exception will be thrown, but the transaction will be cancelled anyway.

Exceptions

<table>
<thead>
<tr>
<th>Xapian::DatabaseError</th>
<th>will be thrown if a problem occurs while modifying the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::DatabaseCorruptError</td>
<td>will be thrown if the database is in a corrupt state.</td>
</tr>
<tr>
<td>Xapian::InvalidOperationError</td>
<td>will be thrown if a transaction is not currently in progress.</td>
</tr>
<tr>
<td>UnimplementedError</td>
<td>will be thrown if transactions are not available for this database type.</td>
</tr>
</tbody>
</table>

7.79.3.6 void Xapian::WritableDatabase::clear_synonyms ( const std::string & term )

Remove all synonyms for a term.

Parameters

<table>
<thead>
<tr>
<th>term</th>
<th>The term to remove all synonyms for. If the term has no synonyms, no action is taken.</th>
</tr>
</thead>
</table>
7.79.3.7 `void Xapian::WritableDatabase::commit();`

Commit any pending modifications made to the database.

For efficiency reasons, when performing multiple updates to a database it is best (indeed, almost essential) to make as many modifications as memory will permit in a single pass through the database. To ensure this, Xapian batches up modifications. This method may be called at any time to commit any pending modifications to the database.

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 `commit()` within a transaction.

Beware of calling `commit()` too frequently: this will make indexing take much longer.

Note that `commit()` 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.

This method was new in Xapian 1.1.0 - in earlier versions it was called `flush()`.

**Exceptions**

<table>
<thead>
<tr>
<th>Xapian::Database-Error</th>
<th>will be thrown if a problem occurs while modifying the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::Database-CorruptError</td>
<td>will be thrown if the database is in a corrupt state.</td>
</tr>
</tbody>
</table>

7.79.3.8 `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**

<table>
<thead>
<tr>
<th>Xapian::Database-Error</th>
<th>will be thrown if a problem occurs while modifying the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::Database-CorruptError</td>
<td>will be thrown if the database is in a corrupt state.</td>
</tr>
</tbody>
</table>
Class Documentation

### 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.

#### 7.79.3.9 void Xapian::WritableDatabase::delete_document ( Xapian::docid did )

Delete a document from the database.

This method removes the document with the specified document ID from the database.

Note that changes to the database won’t be immediately committed to disk; see commit() 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 commit() is called or the database is closed).

**Parameters**

| did  | The document ID of the document to be removed. |

**Exceptions**

<table>
<thead>
<tr>
<th>Xapian::DatabaseError</th>
<th>will be thrown if a problem occurs while writing to the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::DatabaseCorruptError</td>
<td>will be thrown if the database is in a corrupt state.</td>
</tr>
</tbody>
</table>

#### 7.79.3.10 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.

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

[inline]

Pre-1.1.0 name for `commit()`.
Use `commit` instead in new code. This alias may be deprecated in the future.

### 7.79.3.12 void Xapian::WritableDatabase::operator= ( const WritableDatabase & other )

Assignment is allowed.
The internals are reference counted, so assignment is cheap.
Note that only an `WritableDatabase` may be assigned to an `WritableDatabase`: an attempt to assign a `Database` is caught at compile-time.

Parameters

| other | The object to copy. |

### 7.79.3.13 void Xapian::WritableDatabase::remove_spelling ( const std::string & word, Xapian::termcount freqdec = 1 ) const

Remove a word from the spelling dictionary.
The word's frequency is decreased, and if would become zero or less then the word is removed completely.

Parameters

| word | The word to remove. |
| freqdec | How much to decrease its frequency by (default 1). |

### 7.79.3.14 void Xapian::WritableDatabase::remove_synonym ( const std::string & term, const std::string & synonym ) const

Remove a synonym for a term.

Parameters

| term | The term to remove a synonym for. |
| synonym | The synonym to remove. If this isn’t currently a synonym for `term`, then no action is taken. |
7.79.3.15  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 commit() 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 commit() is called or the database is closed).

Parameters

<table>
<thead>
<tr>
<th>did</th>
<th>The document ID of the document to be replaced.</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>The new document.</td>
</tr>
</tbody>
</table>

Exceptions

- Xapian::Database-Error will be thrown if a problem occurs while writing to the database.
- Xapian::Database-CorruptError will be thrown if the database is in a corrupt state.

7.79.3.16  Xapian::docid Xapian::WritableDatabase::replace_document ( const  
std::string & unique_term, const Xapian::Document & document )

Replace any documents matching a term.

This method replaces any documents indexed by the specified term with the specified  
document. If any documents are indexed by the term, the lowest document ID will be  
used for the document, otherwise a new document ID will be generated as for add_document.

One common use is to allow UIDs from another system to easily be mapped to terms in Xapian. Note that this method doesn’t automatically add unique_term as a term,  
so you’ll need to call document.add_term(unique_term) first when using replace_document() in this way.

Note that changes to the database won’t be immediately committed to disk; see commit() 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 commit() is called or the database is closed).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unique_term</td>
<td>The “unique” term.</td>
</tr>
<tr>
<td>document</td>
<td>The new document.</td>
</tr>
</tbody>
</table>

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.

7.79.3.17 void Xapian::WritableDatabase::set_metadata ( const std::string & key, const std::string & value )

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

This method sets the metadata value associated with a given key. If there is already a metadata value stored in the database with the same key, the old value is replaced. If you want to delete an existing item of metadata, just set its value to the empty string.

User-specified metadata allows you to store arbitrary information in the form of (key,tag) pairs.

There’s no hard limit on the number of metadata items, or the size of the metadata values. Metadata keys have a limited length, which depends on the backend. We recommend limiting them to 200 bytes. Empty keys are not valid, and specifying one will cause an exception.

Metadata modifications are committed to disk in the same way as modifications to the documents in the database are: i.e., modifications are atomic, and won’t be committed to disk immediately (see commit() 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>The key of the metadata item to set.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the metadata item to set.</td>
</tr>
</tbody>
</table>
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xapian::DatabaseError</td>
<td>Will be thrown if a problem occurs while writing to the database.</td>
</tr>
<tr>
<td>Xapian::DatabaseCorruptError</td>
<td>Will be thrown if the database is in a corrupt state.</td>
</tr>
<tr>
<td>Xapian::InvalidArgumentError</td>
<td>Will be thrown if the key supplied is empty.</td>
</tr>
<tr>
<td>Xapian::UnimplementedError</td>
<td>Will be thrown if the database backend in use doesn't support user-specified metadata.</td>
</tr>
</tbody>
</table>

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

- xapian/database.h
Chapter 8

File Documentation

8.1 xapian/error.h File Reference

Hierarchy of classes which Xapian can throw as exceptions.

Classes

- class Xapian::Error
  
  All exceptions thrown by Xapian are subclasses of Xapian::Error.

- class Xapian::LogicError
  
  The base class for exceptions indicating errors in the program logic.

- class Xapian::RuntimeError
  
  The base class for exceptions indicating errors only detectable at runtime.

- class Xapian::AssertionError
  
  AssertionError is thrown if a logical assertion inside Xapian fails.

- class Xapian::InvalidArgumentError
  
  InvalidArgumentError indicates an invalid parameter value was passed to the API.

- class Xapian::InvalidOperationError
  
  InvalidOperationError indicates the API was used in an invalid way.

- class Xapian::UnimplementedError
  
  UnimplementedError indicates an attempt to use an unimplemented feature.

- class Xapian::DatabaseError
  
  DatabaseError indicates some sort of database related error.

- class Xapian::DatabaseCorruptError
  
  DatabaseCorruptError indicates database corruption was detected.

- class Xapian::DatabaseCreateError
  
  DatabaseCreateError indicates a failure to create a database.

- class Xapian::DatabaseLockError
  
  DatabaseLockError indicates failure to lock a database.
• class Xapian::DatabaseModifiedError
  
  *DatabaseModifiedError* indicates a database was modified.

• class Xapian::DatabaseOpeningError
  
  *DatabaseOpeningError* indicates failure to open a database.

• class Xapian::DatabaseVersionError
  
  *DatabaseVersionError* indicates that a database is in an unsupported format.

• class Xapian::DocNotFoundError
  
  Indicates an attempt to access a document not present in the database.

• class Xapian::FeatureUnavailableError
  
  Indicates an attempt to use a feature which is unavailable.

• class Xapian::InternalError
  
  *InternalError* indicates a runtime problem of some sort.

• class Xapian::NetworkError
  
  Indicates a problem communicating with a remote database.

• class Xapian::NetworkTimeoutError
  
  Indicates a timeout expired while communicating with a remote database.

• class Xapian::QueryParserError
  
  Indicates a query string can't be parsed.

• class Xapian::SerialisationError
  
  Indicates an error in the std::string serialisation of an object.

• class Xapian::RangeError
  
  *RangeError* indicates an attempt to access outside the bounds of a container.

Namespaces

• namespace Xapian
  
  The *Xapian* namespace contains public interfaces for the *Xapian* library.

8.1.1 Detailed Description

Hierarchy of classes which *Xapian* can throw as exceptions.

8.2 xapian/version.h File Reference

Define preprocessor symbols for the library version.

Defines

• #define XAPIAN_ENABLE_VISIBILITY
  
  The library was compiled with GCC's -fvisibility=hidden option.

• #define XAPIAN_VERSION "1.3.1"
8.3 xapian.h File Reference

The version of Xapian as a C string literal.

- \#define XAPIAN_MAJOR_VERSION 1
  The major component of the Xapian version.

- \#define XAPIAN_MINOR_VERSION 3
  The minor component of the Xapian version.

- \#define XAPIAN_REVISION 1
  The revision component of the Xapian version.

- \#define XAPIAN_HAS_BRASS_BACKEND 1
  XAPIAN_HAS_BRASS_BACKEND Defined if the brass backend is enabled.

- \#define XAPIAN_HAS_CHERT_BACKEND 1
  XAPIAN_HAS_CHERT_BACKEND Defined if the chert backend is enabled.

- \#define XAPIAN_HAS_INMEMORY_BACKEND 1
  XAPIAN_HAS_INMEMORY_BACKEND Defined if the inmemory backend is enabled.

- \#define XAPIAN_HAS_REMOTE_BACKEND 1
  XAPIAN_HAS_REMOTE_BACKEND Defined if the remote backend is enabled.

8.2.1 Detailed Description

Define preprocessor symbols for the library version.

8.2.2 Define Documentation

8.2.2.1 \#define XAPIAN_MAJOR_VERSION 1

The major component of the Xapian version.
E.g. for Xapian 1.0.14 this would be: 1

8.2.2.2 \#define XAPIAN_MINOR_VERSION 3

The minor component of the Xapian version.
E.g. for Xapian 1.0.14 this would be: 0

8.2.2.3 \#define XAPIAN_REVISION 1

The revision component of the Xapian version.
E.g. for Xapian 1.0.14 this would be: 14

8.3 xapian.h File Reference

Public interfaces for the Xapian library.
Namespaces

- namespace Xapian

  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- const char * Xapian::version_string ()

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

- int Xapian::major_version ()

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

- int Xapian::minor_version ()

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

- int Xapian::revision ()

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

8.3.1 Detailed Description

Public interfaces for the Xapian library.

8.4 xapian/attributes.h File Reference

Compiler attribute macros.

Defines

- #define XAPIAN_CONST_FUNCTION

  A function which does not examine any values except its arguments and has no effects except its return value.

- #define XAPIAN_PURE_FUNCTION

  Like XAPIAN_CONST_FUNCTION, but such a function can also examine global memory, perhaps via pointer or reference parameters.

8.4.1 Detailed Description

Compiler attribute macros.
8.4.2 Define Documentation

8.4.2.1 `#define XAPIAN_CONST_FUNCTION`

A function which does not examine any values except its arguments and has no effects except its return value.

This means the compiler can perform CSE (common subexpression elimination) on calls to such a function with the same arguments, and also completely eliminate calls to this function when the return value isn't used.

8.5 `xapian/compactor.h` File Reference

Compact a database, or merge and compact several.

Classes

- class `Xapian::Compactor`
  
  *Compact a database, or merge and compact several.*

Namespaces

- namespace `Xapian`
  
  *The `Xapian` namespace contains public interfaces for the `Xapian` library.*

8.5.1 Detailed Description

Compact a database, or merge and compact several.

8.6 `xapian/database.h` File Reference

API for working with `Xapian` databases.

Classes

- class `Xapian::Database`
  
  *This class is used to access a database, or a group of databases.*

- class `Xapian::WritableDatabase`
  
  *This class provides read/write access to a database.*
Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

Variables

- const int Xapian::DB_CREATE_OR_OPEN = 1
  
  Open for read/write; create if no db exists.

- const int Xapian::DB_CREATE = 2
  
  Create a new database; fail if db exists.

- const int Xapian::DB_CREATE_OR_OVERWRITE = 3
  
  Overwrite existing db; create if none exists.

- const int Xapian::DB_OPEN = 4
  
  Open for read/write; fail if no db exists.

- const int Xapian::DBCHECK_SHORT_TREE = 1
  
  Show a short-format display of the B-tree contents.

- const int Xapian::DBCHECK_FULL_TREE = 2
  
  Show a full display of the B-tree contents.

- const int Xapian::DBCHECK_SHOW_BITMAP = 4
  
  Show the bitmap for the B-tree.

- const int Xapian::DBCHECK_SHOW_STATS = 8
  
  Show statistics for the B-tree.

- const int Xapian::DBCHECK_FIX = 16
  
  Fix problems.

8.6.1 Detailed Description

API for working with Xapian databases.

8.7 xapian/dbfactory.h File Reference

Factory functions for constructing Database and WritableDatabase objects.

Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

- namespace Xapian::Auto
  
  Database factory functions which determine the database type automatically.

- namespace Xapian::InMemory
  
  Database factory functions for the inmemory backend.
8.7 xapian/dbfactory.h File Reference

- namespace Xapian::Brass
  Database factory functions for the brass backend.
- namespace Xapian::Chert
  Database factory functions for the chert backend.
- namespace Xapian::Remote
  Database factory functions for the remote backend.

Functions

- Database Xapian::Auto::open_stub (const std::string &file)
  Construct a Database object for a stub database file.
- WritableDatabase Xapian::Auto::open_stub (const std::string &file, int action)
  Construct a WritableDatabase object for a stub database file.
- WritableDatabase Xapian::InMemory::open ()
  Construct a WritableDatabase object for a new, empty InMemory database.
- Database Xapian::Brass::open (const std::string &dir)
  Construct a Database object for read-only access to a Brass database.
- WritableDatabase Xapian::Brass::open (const std::string &dir, int action, int block_size=8192)
  Construct a Database object for update access to a Brass database.
- Database Xapian::Chert::open (const std::string &dir)
  Construct a Database object for read-only access to a Chert database.
- WritableDatabase Xapian::Chert::open (const std::string &dir, int action, int block_size=8192)
  Construct a Database object for update access to a Chert database.
- Database Xapian::Remote::open (const std::string &host, unsigned int port, useconds_t timeout=10000, useconds_t connect_timeout=10000)
  Construct a Database object for read-only access to a remote database accessed via a TCP connection.
- WritableDatabase Xapian::Remote::open_writable (const std::string &host, unsigned int port, useconds_t timeout=0, useconds_t connect_timeout=10000)
  Construct a WritableDatabase object for update access to a remote database accessed via a TCP connection.
- Database Xapian::Remote::open (const std::string &program, const std::string &args, useconds_t timeout=10000)
  Construct a Database object for read-only access to a remote database accessed via a program.
- WritableDatabase Xapian::Remote::open_writable (const std::string &program, const std::string &args, useconds_t timeout=0)
  Construct a WritableDatabase object for update access to a remote database accessed via a program.

8.7.1 Detailed Description

Factory functions for constructing Database and WritableDatabase objects.
8.8  xapian/document.h File Reference

API for working with documents.

Classes

• class Xapian::Document
  A handle representing a document in a Xapian database.

Namespaces

• namespace Xapian
  The Xapian namespace contains public interfaces for the Xapian library.

8.8.1 Detailed Description

API for working with documents.

8.9  xapian/enquire.h File Reference

API for running queries.

Classes

• class Xapian::MSet
  A match set (MSet).
• class Xapian::MSetIterator
  An iterator pointing to items in an MSet.
• class Xapian::ESet
  Class representing an ordered set of expand terms (an ESet).
• class Xapian::ESetIterator
  Iterate through terms in the ESet.
• class Xapian::RSet
  A relevance set (R-Set).
• class Xapian::MatchDecider
  Base class for matcher decision functor.
• class Xapian::Enquire
  This class provides an interface to the information retrieval system for the purpose of searching.
Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const MSetIterator &a, const MSetIterator &b)
  Equality test for MSetIterator objects.
- bool Xapian::operator!= (const MSetIterator &a, const MSetIterator &b)
  Inequality test for MSetIterator objects.
- bool Xapian::operator== (const ESetIterator &a, const ESetIterator &b)
  Equality test for ESetIterator objects.
- bool Xapian::operator!= (const ESetIterator &a, const ESetIterator &b)
  Inequality test for ESetIterator objects.

8.9.1 Detailed Description

API for running queries.

8.10 xapian/errorhandler.h File Reference

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

Classes

- class Xapian::ErrorHandler
  
  Decide if a Xapian::Error exception should be ignored.

Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

8.10.1 Detailed Description

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

8.11 xapian/expanddecider.h File Reference

Allow rejection of terms during ESet generation.
Classes

- class **Xapian::ExpandDecider**
  Virtual base class for expand decider functor.
- class **Xapian::ExpandDeciderAnd**
  ExpandDecider subclass which rejects terms using two ExpandDeciders.
- class **Xapian::ExpandDeciderFilterTerms**
  ExpandDecider subclass which rejects terms in a specified list.
- class **Xapian::ExpandDeciderFilterPrefix**
  ExpandDecider subclass which restrict terms to a particular prefix.

Namespaces

- namespace **Xapian**
  The Xapian namespace contains public interfaces for the Xapian library.

8.11.1 Detailed Description

Allow rejection of terms during ESet generation.

8.12 xapian/geospatial.h File Reference

Geospatial search support routines.

Classes

- struct **Xapian::LatLongCoord**
  A latitude-longitude coordinate.
- class **Xapian::LatLongCoordsIterator**
  An iterator across the values in a LatLongCoords object.
- class **Xapian::LatLongCoords**
  A sequence of latitude-longitude coordinates.
- class **Xapian::LatLongMetric**
  Base class for calculating distances between two lat/long coordinates.
- class **Xapian::GreatCircleMetric**
  Calculate the great-circle distance between two coordinates on a sphere.
- class **Xapian::LatLongDistancePostingSource**
  Posting source which returns a weight based on geospatial distance.
- class **Xapian::LatLongDistanceKeyMaker**
  KeyMaker subclass which sorts by distance from a latitude/longitude.
8.13 xapian/keymaker.h File Reference

Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- double Xapian::miles_to_metres (double miles)
  
  Convert from miles to metres.
- double Xapian::metres_to_miles (double metres)
  
  Convert from metres to miles.
- bool Xapian::operator!= (const LatLongCoordsIterator &a, const LatLongCoordsIterator &b)
  
  Inequality test for LatLongCoordsIterator objects.

8.12.1 Detailed Description

Geospatial search support routines.

8.13 xapian/keymaker.h File Reference

Build key strings for MSet ordering or collapsing.

Classes

- class Xapian::KeyMaker
  
  Virtual base class for key making functors.
- class Xapian::MultiValueKeyMaker
  
  KeyMaker subclass which combines several values.

Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

8.13.1 Detailed Description

Build key strings for MSet ordering or collapsing.
8.14 xapian/matchspy.h File Reference

MatchSpy implementation.

Classes

- class Xapian::MatchSpy
  Abstract base class for match spies.
- class Xapian::ValueCountMatchSpy
  Class for counting the frequencies of values in the matching documents.

Namespaces

- namespace Xapian
  The Xapian namespace contains public interfaces for the Xapian library.

8.14.1 Detailed Description

MatchSpy implementation.

8.15 xapian/positioniterator.h File Reference

Class for iterating over term positions.

Classes

- class Xapian::PositionIterator
  Class for iterating over term positions.

Namespaces

- namespace Xapian
  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const PositionIterator &a, const PositionIterator &b)
  Equality test for PositionIterator objects.
- bool Xapian::operator!= (const PositionIterator &a, const PositionIterator &b)
  Inequality test for PositionIterator objects.
8.15.1 Detailed Description

Class for iterating over term positions.

8.16 xapian/postingiterator.h File Reference

Class for iterating over a list of document ids.

Classes

- class Xapian::PostingIterator
  
  Class for iterating over a list of terms.

Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- bool Xapian::operator== (const PostingIterator &a, const PostingIterator &b)
  
  Equality test for PostingIterator objects.

- bool Xapian::operator!= (const PostingIterator &a, const PostingIterator &b)
  
  Inequality test for PostingIterator objects.

8.16.1 Detailed Description

Class for iterating over a list of document ids.

8.17 xapian/postingsource.h File Reference

External sources of posting information.

Classes

- class Xapian::PostingSource
  
  Base class which provides an "external" source of postings.

- class Xapian::ValuePostingSource
  
  A posting source which generates weights from a value slot.

- class Xapian::ValueWeightPostingSource
A posting source which reads weights from a value slot.

- class `Xapian::DecreasingValueWeightPostingSource`
  
  Read weights from a value which is known to decrease as docid increases.

- class `Xapian::ValueMapPostingSource`
  
  A posting source which looks up weights in a map using values as the key.

- class `Xapian::FixedWeightPostingSource`
  
  A posting source which returns a fixed weight for all documents.

Namespaces

- namespace `Xapian`
  
  The `Xapian` namespace contains public interfaces for the `Xapian` library.

8.17.1 Detailed Description

External sources of posting information.

8.18 xapian/query.h File Reference

`Xapian::Query` API class.

Classes

- class `Xapian::Query`
  
  Class representing a query.

Namespaces

- namespace `Xapian`
  
  The `Xapian` namespace contains public interfaces for the `Xapian` library.

8.18.1 Detailed Description

`Xapian::Query` API class.

8.19 xapian/queryparser.h File Reference

parsing a user query string to build a `Xapian::Query` object
Classes

- **class Xapian::Stopper**
  Base class for stop-word decision functor.
- **class Xapian::SimpleStopper**
  Simple implementation of Stopper class - this will suit most users.
- **struct Xapian::ValueRangeProcessor**
  Base class for value range processors.
- **class Xapian::StringValueRangeProcessor**
  Handle a string range.
- **class Xapian::DateValueRangeProcessor**
  Handle a date range.
- **class Xapian::NumberValueRangeProcessor**
  Handle a number range.
- **struct Xapian::FieldProcessor**
  Base class for field processors.
- **class Xapian::QueryParser**
  Build a Xapian::Query object from a user query string.

Namespaces

- **namespace Xapian**
  The Xapian namespace contains public interfaces for the Xapian library.

Functions

- **std::string Xapian::sortable_serialise (double value)**
  Convert a floating point number to a string, preserving sort order.
- **double Xapian::sortable_unserialise (const std::string &value)**
  Convert a string encoded using sortable_serialise back to a floating point number.

8.19.1 Detailed Description

parsing a user query string to build a Xapian::Query object

8.20 xapian/registry.h File Reference

Class for looking up user subclasses during unserialization.
Classes

• class Xapian::Registry
  Registry for user subclasses.

Namespaces

• namespace Xapian
  The Xapian namespace contains public interfaces for the Xapian library.

8.20.1 Detailed Description

Class for looking up user subclasses during unserialisation.

8.21 xapian/stem.h File Reference

stemming algorithms

Classes

• struct Xapian::StemImplementation
  Class representing a stemming algorithm implementation.
• class Xapian::Stem
  Class representing a stemming algorithm.

Namespaces

• namespace Xapian
  The Xapian namespace contains public interfaces for the Xapian library.

8.21.1 Detailed Description

stemming algorithms

8.22 xapian/termgenerator.h File Reference

parse free text and generate terms
Classes

- class `Xapian::TermGenerator`
  
  Parses a piece of text and generate terms.

Namespaces

- namespace `Xapian`

  The `Xapian` namespace contains public interfaces for the `Xapian` library.

8.22.1 Detailed Description

parse free text and generate terms

8.23 xapian/termiterator.h File Reference

Class for iterating over a list of terms.

Classes

- class `Xapian::TermIterator`
  
  Class for iterating over a list of terms.

Namespaces

- namespace `Xapian`

  The `Xapian` namespace contains public interfaces for the `Xapian` library.

Functions

- bool `Xapian::operator==` (const TermIterator &a, const TermIterator &b)
  
  Equality test for `TermIterator` objects.

- bool `Xapian::operator!=` (const TermIterator &a, const TermIterator &b)
  
  Inequality test for `TermIterator` objects.

8.23.1 Detailed Description

Class for iterating over a list of terms.
typedefs for Xapian

Namespaces

• namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

Typedefs

• typedef unsigned Xapian::doccount
  A count of documents.
• typedef int Xapian::doccount_diff
  A signed difference between two counts of documents.
• typedef unsigned Xapian::docid
  A unique identifier for a document.
• typedef double Xapian::doclength
  A normalised document length.
• typedef int Xapian::percent
  The percentage score for a document in an MSet.
• typedef unsigned Xapian::termcount
  A counts of terms.
• typedef int Xapian::termcount_diff
  A signed difference between two counts of terms.
• typedef unsigned Xapian::termpos
  A term position within a document or query.
• typedef int Xapian::termpos_diff
  A signed difference between two term positions.
• typedef unsigned Xapian::timeout
  A timeout value in milliseconds.
• typedef unsigned Xapian::valueno
  The number for a value slot in a document.
• typedef int Xapian::valueno_diff
  A signed difference between two value slot numbers.
• typedef double Xapian::weight
  The weight of a document or term.

Variables

• const valueno Xapian::BAD_VALUENO = 0xffffffff
  Reserved value to indicate "no valueno".
8.24.1 Detailed Description

typedefs for Xapian

8.25 xapian/unicode.h File Reference

Unicode and UTF-8 related classes and functions.

Classes

• class Xapian::Utf8Iterator
  An iterator which returns Unicode character values from a UTF-8 encoded string.

Namespaces

• namespace Xapian
  The Xapian namespace contains public interfaces for the Xapian library.
• namespace Xapian::Unicode
  Functions associated with handling Unicode characters.

Enumerations

• enum Xapian::Unicode::category
  Each Unicode character is in exactly one of these categories.

Functions

• unsigned Xapian::Unicode::nonascii_to_utf8 (unsigned ch, char ∗buf)
  Convert a single non-ASCII Unicode character to UTF-8.
• unsigned Xapian::Unicode::to_utf8 (unsigned ch, char ∗buf)
  Convert a single Unicode character to UTF-8.
• void Xapian::Unicode::append_utf8 (std::string ∗s, unsigned ch)
  Append the UTF-8 representation of a single Unicode character to a std::string.
• category Xapian::Unicode::get_category (unsigned ch)
  Return the category which a given Unicode character falls into.
• bool Xapian::Unicode::is_wordchar (unsigned ch)
  Test if a given Unicode character is "word character".
• bool Xapian::Unicode::is_whitespace (unsigned ch)
  Test if a given Unicode character is a whitespace character.
• bool Xapian::Unicode::is_currency (unsigned ch)
  Test if a given Unicode character is a currency symbol.
• unsigned Xapian::Unicode::tolower (unsigned ch)
  Convert a Unicode character to lowercase.
• unsigned Xapian::Unicode::toupper (unsigned ch)
  Convert a Unicode character to uppercase.
• std::string Xapian::Unicode::tolower (const std::string &term)
  Convert a UTF-8 std::string to lowercase.
• std::string Xapian::Unicode::toupper (const std::string &term)
  Convert a UTF-8 std::string to uppercase.

8.25.1 Detailed Description

Unicode and UTF-8 related classes and functions.

8.26 xapian/valueiterator.h File Reference

Class for iterating over document values.

Classes

• class Xapian::ValueIterator
  Class for iterating over document values.

Namespaces

• namespace Xapian
  The Xapian namespace contains public interfaces for the Xapian library.

Functions

• bool Xapian::operator== (const ValueIterator &a, const ValueIterator &b)
  Equality test for ValueIterator objects.
• bool Xapian::operator!= (const ValueIterator &a, const ValueIterator &b)
  Inequality test for ValueIterator objects.

8.26.1 Detailed Description

Class for iterating over document values.

8.27 xapian/valuesetmatchdecider.h File Reference

MatchDecider subclass for filtering results by value.
8.28 xapian/weight.h File Reference

Classes

- class Xapian::ValueSetMatchDecider
  
  MatchDecider filtering results based on whether document values are in a user-defined set.

Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

8.27.1 Detailed Description

MatchDecider subclass for filtering results by value.

8.28 xapian/weight.h File Reference

Weighting scheme API.

Classes

- class Xapian::Weight
  
  Abstract base class for weighting schemes.
- class Xapian::BoolWeight
  
  Class implementing a "boolean" weighting scheme.
- class Xapian::TfidfWeight
  
  Xapian::Weight subclass implementing the tf-idf weighting scheme.
- class Xapian::BM25Weight
  
  Xapian::Weight subclass implementing the BM25 probabilistic formula.
- class Xapian::TradWeight
  
  Xapian::Weight subclass implementing the traditional probabilistic formula.

Namespaces

- namespace Xapian
  
  The Xapian namespace contains public interfaces for the Xapian library.

8.28.1 Detailed Description

Weighting scheme API.