Xqueeze RI C++ Reference Manual

Tahir Hashmi

March 18, 2003

Contents

1 Xqueeze Reference Implementation (C++) for xqML Processing **3**

1.1 Introduction . . . 3

1.2 Copying . . . 3

2 Module Documentation **3**

2.1 Exceptions . . . 3

2.1.1 Detailed Description . . . 3

2.2 Xqueeze Parser . . . 4

2.2.1 Detailed Description . . . 4

2.3 Xqueeze Writer . . . 4

2.3.1 Detailed Description . . . 4

2.4 Xqueeze Association Generator . . . 4

2.4.1 Detailed Description . . . 4

2.5 xqML Symbols . . . 4

3 Namespace Documentation **5**

3.1 xqML Namespace Reference . . . 5

3.1.1 Detailed Description . . . 5

3.1.2 Function Documentation . . . 5

3.2 xqMLParser Namespace Reference . . . 6

3.2.1 Detailed Description . . . 6

3.2.2 Enumeration Type Documentation . . . 6

3.3 xqMLWriter Namespace Reference . . . 7

3.3.1 Detailed Description . . . 7

3.3.2 Enumeration Type Documentation . . . 7

3.4 xqMLxqAGen Namespace Reference . . . 8

3.4.1 Detailed Description . . . 8

4 Class Documentation **8**

4.1 xqML::IllegalContext Struct Reference 8

4.1.1 Detailed Description 9

4.1.2 Constructor & Destructor Documentation 9

4.2 xqML::IllegalIdentifier Struct Reference 10

4.2.1 Detailed Description 10

4.2.2 Constructor & Destructor Documentation 10

4.3 xqMLWriter::Mapping Class Reference 11

4.3.1 Detailed Description 11

4.3.2 Constructor & Destructor Documentation 11

4.4 xqMLParser::Mapping Class Reference 12

4.4.1 Detailed Description 12

4.4.2 Constructor & Destructor Documentation 12

4.5 xqML::Parser Class Reference 12

4.5.1 Detailed Description 13

4.5.2 Constructor & Destructor Documentation 13

4.6 xqMLParser::ParserFramework Class Reference 14

4.6.1 Detailed Description 14

4.7 xqML::Writer Class Reference 14

4.7.1 Detailed Description 14

4.7.2 Constructor & Destructor Documentation 15

4.7.3 Member Function Documentation 16

4.8 xqMLWriter::WriterFramework Class Reference 18

4.8.1 Detailed Description 19

4.9 xqML::xqAGenerator Class Reference 19

4.9.1 Detailed Description 19

4.9.2 Constructor & Destructor Documentation 19

4.9.3 Member Function Documentation 19

4.10 xqMLxqAGen::xqAGeneratorFramework Class Reference 20

4.10.1 Detailed Description 20

4.11 xqML::xqMLSymbol Class Reference 20

4.11.1 Detailed Description 20

4.11.2 Friends And Related Function Documentation 21

5 GNU Free Documentation License **21**

5.1 Applicability and Definitions 21

5.2 Verbatim Copying 22

5.3 Copying in Quantity 22
5.4 Modifications 22
5.5 Combining Documents 23
5.6 Collections of Documents 23
5.7 Aggregation with Independent Works 23
5.8 Translation 24
5.9 Termination 24
5.10 Future Revisions of This License 24

1 Xqueeze Reference Implementation (C++) for xqML Processing

1.1 Introduction

Xqueeze Reference Implementation is an implementation of the Xqueeze API for xqML generation and parsing. This document describes the Xqueeze API as well as contains Source Documentation. The API is described in the “compounds” linked from the section entitled Module Documentation

1.2 Copying

2 Module Documentation

2.1 Exceptions

2.1.1 Detailed Description

These are the exceptions thrown by the modules in Xqueeze.

Compounds

- struct IllegalContext

  Exception thrown due to bad document structure.

- struct IllegalIdentifier

  Exception thrown when lookup for an identifier or symbol fails.

2.2 Xqueeze Parser

2.2.1 Detailed Description

xqML parsing API.

Xqueeze Parser interprets an xqML document and writes an XML-like equivalent on the desired output stream. The parser does not require an xqA specification for parsing but it won't be able to translate the symbols to their literal identifiers.

Compounds

- class Parser

  Provides API to the client for parsing xqML Documents.

2.3 Xqueeze Writer

2.3.1 Detailed Description

xqML Generation API

Xqueeze Writer provides API for generation of xqML documents against a given xqA specification. The API allows generation of xqML documents through a set of SAX-like calls that write various XML structures in xqML encoding.

Compounds

- class Writer

  Provides API to the client for generation of xqML Documents.

2.4 Xqueeze Association Generator

2.4.1 Detailed Description

API for generating xqA specifications.

This is used to generate xqA specification against a DTD for generation of xqML documents against it.

Compounds

- class xqAGenerator

  This class is used for creating an xqA out of a DTD.

2.5 xqML Symbols

Compounds

- class xqMLSymbol

Provides a data type for xqML Symbols and I/O facilities.

3 Namespace Documentation

3.1 xqML Namespace Reference

3.1.1 Detailed Description

Namespace for client accessible interfaces.

Compounds

- struct IllegalContext

  Exception thrown due to bad document structure.

- struct IllegalIdentifier

  Exception thrown when lookup for an identifier or symbol fails.

- class Parser

  Provides API to the client for parsing xqML Documents.

- class Writer

  Provides API to the client for generation of xqML Documents.

- class xqAGenerator

  This class is used for creating an xqA out of a DTD.

- class xqMLSymbol

  Provides a data type for xqML Symbols and I/O facilities.

Functions

- ostream & operator<< (ostream &stream, xqMLSymbol symbol)
- istream & operator>> (istream &stream, xqMLSymbol &symbol)

3.1.2 Function Documentation

3.1.2.1 ostream& operator<< (ostream & *stream*, xqMLSymbol *symbol*)

This operator is capable of writing any unsigned integer in network byte order since it doesn't check whether the xqMLSymbol is indeed valid (i.e. properly uses the lsb as a continuation flag)

3.1.2.2 istream& operator>> (istream & *stream*, xqMLSymbol & *symbol*)

Can't handle an xqMLSymbol whose value is greater than the largest value an unsigned integer can hold on the platform used.

3.2 xqMLParser Namespace Reference

3.2.1 Detailed Description

Namespace for implementation specific interfaces for Parser.

Compounds

- class Mapping

  Support class that reads an Xqueeze Mapping specification and provides symbol-to-name/type lookup service for that specification.

- class ParserFramework

  Class to hold private members for Writer.

Enumerations

- enum Context { prolog, start_tag, ee_start_tag, element, open_attribute, attribute_w_predef_value }

  Document contexts for tracking the structure of the document.

- enum xqMLType { EL, EE, AT, AP, VA, EN, NS }

  Types of xqML Symbols.

Functions

- string **contextName** (Context)

3.2.2 Enumeration Type Documentation

3.2.2.1 enum xqMLParser::Context

Document contexts for tracking the structure of the document.

This is used by WriterFramework to keep track of the structure of the document being produced.

Enumeration values:

prolog Document prolog.

start_tag (element) start tag

ee_start_tag empty element start tag

element element

open_attribute open attribute

attribute_w_predef_value open attribute with predefined value

3.2.2.2 enum xqMLParser::xqMLType

Types of xqML Symbols.

This is used by Mapping to identify the types of strings for servicing symbol lookup requests.

Enumeration values:

EL element

EE empty element

AT attribute

AP attribute with predefined value

VA predefined attribute value

EN entity reference

NS namespace

3.3 xqMLWriter Namespace Reference

3.3.1 Detailed Description

Namespace for implementation specific interfaces for Writer.

Compounds

- class Mapping

  Support class that reads an Xqueeze Mapping specification and provides name-to-symbol lookup service for that specification.

- class WriterFramework

  Class to hold private members for Writer.

Enumerations

- enum Context { preamble, start_tag, ee_start_tag, element, open_attribute, attribute_w_predef_value }

  Document contexts for tracking the structure of the document.

- enum xqMLType { EL, EE, AT, AP, VA, EN, NS }

  Types of xqML Symbols.

3.3.2 Enumeration Type Documentation

3.3.2.1 enum xqMLWriter::Context

Document contexts for tracking the structure of the document.

This is used by WriterFramework to keep track of the structure of the document being produced.

Enumeration values:

preamble Document preamble.

start_tag (element) start tag

ee_start_tag empty element start tag

element element

open_attribute open attribute

attribute_w_predef_value open attribute with predefined value

3.3.2.2 enum xqMLWriter::xqMLType

Types of xqML Symbols.

This is used by Mapping to identify the types of strings for servicing symbol lookup requests.

Enumeration values:

EL element

EE empty element

AT attribute

AP attribute with predefined value

VA predefined attribute value

EN entity reference

NS namespace

3.4 xqMLxqAGen Namespace Reference

3.4.1 Detailed Description

Namespace for implementation specific interfaces for xqAGenerator.

Compounds

- class xqAGeneratorFramework

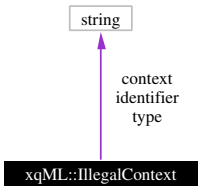
  Class to hold private members for xqAGenerator.

4 Class Documentation

4.1 xqML::IllegalContext Struct Reference

`#include <xqml/exceptions.h>`

Collaboration diagram for xqML::IllegalContext:



4.1.1 Detailed Description

Exception thrown due to bad document structure.

This exception is thrown by Writer methods if the request is not possible to be serviced since it violates the correct xqML document structure. The Parser throws this exception when it encounters malformed xqML structure.

Public Member Functions

- IllegalContext (const string &ill_context, const string &id, const string &id_type)

  Gives context, name of identifier and type of identifier where bad structure was found.

- IllegalContext (const string &ill_context, const unsigned int code)

  Gives context and decimal value (as string) of xqML Symbol that caused bad structure.

Public Attributes

- string context

  name of the current context of the document

- string identifier

  name of the offending identifier

- string type

  type name of the identifier

4.1.2 Constructor & Destructor Documentation

4.1.2.1 xqML::IllegalContext::IllegalContext (const string & *ill_context*, const string & *id*, const string & *id_type*) [inline]

Gives context, name of identifier and type of identifier where bad structure was found.

Parameters:

ill_context Name of the document context where exception occured

id Name of the identifier that caused the exception

id_type Type of the identifier that caused the exception

4.1.2.2 xqML::IllegalContext::IllegalContext (const string & *ill_context*, const unsigned int *code*) [inline]

Gives context and decimal value (as string) of xqML Symbol that caused bad structure.

Parameters:

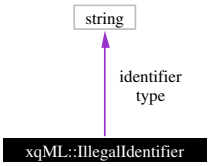
ill_context Name of the document context where exception occured

code Value of xqML Symbol that caused the exception

4.2 xqML::IllegalIdentifier Struct Reference

`#include <xqml/exceptions.h>`

Collaboration diagram for xqML::IllegalIdentifier:



4.2.1 Detailed Description

Exception thrown when lookup for an identifier or symbol fails.

Public Member Functions

- IllegalIdentifier (const string &id, const string &id_type)
  Gives name and type of identifier for which a matching symbol was not found.
- IllegalIdentifier (const unsigned int code)
  Gives decimal value of xqML Symbol (as string) that could not.

Public Attributes

- string identifier
  indentifier that could not be found
- string type
  type name of the identifier

4.2.2 Constructor & Destructor Documentation

4.2.2.1 xqML::IllegalIdentifier::IllegalIdentifier (const string & *id*, const string & *id_type*) `[inline]`

Gives name and type of identifier for which a matching symbol was not found.

Parameters:

id Name of the identifier for which a symbol could not be found

id_type Type of such identifier

4.2.2.2 xqML::IllegalIdentifier::IllegalIdentifier (const unsigned int *code*) [inline]

Gives decimal value of xqML Symbol (as string) that could not.

Parameters:

code Value of xqML Symbol that could not be translated

4.3 xqMLWriter::Mapping Class Reference

`#include <writer_impl.h>`

4.3.1 Detailed Description

Support class that reads an Xqueeze Mapping specification and provides name-to-symbol lookup service for that specification.

Public Member Functions

- Mapping (istream &)

  Constructor reads specifications from istream argument and builds a lookup table.

- xqMLSymbol GetEmptyElement (const string &)

  Return xqMLSymbol of empty element given as argument.

- xqMLSymbol GetElement (const string &)

  Return xqMLSymbol of element given as argument.

- xqMLSymbol GetAttribute (const string &)

  Return xqMLSymbol of attribute given as argument.

- xqMLSymbol GetAttributeWPV (const string &)

  Return xqMLSymbol of attribute with predefined values given as argument.

- xqMLSymbol GetAttributeValue (const string &)

  Return xqMLSymbol of predefined attribute value given as argument.

- xqMLSymbol GetEntityReference (const string &)

  Return xqMLSymbol of entity reference given as argument.

- xqMLSymbol GetNamespace (const string &)

  Return xqMLSymbol of namespace given as argument.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 xqMLWriter::Mapping::Mapping (istream & *source_file*) [explicit]

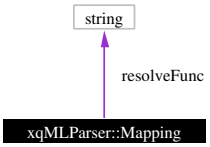
Constructor reads specifications from istream argument and builds a lookup table.

Constructor calls buildMap()

4.4 xqMLParser::Mapping Class Reference

`#include <parser_impl.h>`

Collaboration diagram for xqMLParser::Mapping:



4.4.1 Detailed Description

Support class that reads an Xqueeze Mapping specification and provides symbol-to-name/type lookup service for that specification.

Public Member Functions

- Mapping (istream &)

  Constructor (translation enabled).

- Mapping ()

  Constructor (translation disabled).

- string getName (unsigned int code)

  Returns name or decimal representation of the value of xqML Symbol passed as argument.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 xqMLParser::Mapping::Mapping (istream & *source_file*) `[explicit]`

Constructor (translation enabled).

This constructor enables translation of symbol values to their names according to the xqA spec given as argument. Calls buildMap() to construct a lookup table and initializes resolveFunc to getNameString.

4.4.2.2 xqMLParser::Mapping::Mapping ()

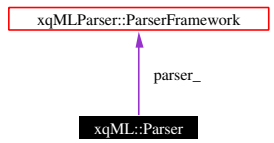
Constructor (translation disabled).

This constructor disables translation of symbol values to their names. Initializes resolveFunc to getDecimalValue

4.5 xqML::Parser Class Reference

`#include <xqml/parser.h>`

Collaboration diagram for xqML::Parser:



4.5.1 Detailed Description

Provides API to the client for parsing xqML Documents.

Public Member Functions

- Parser (istream &, istream &, ostream &)

  Constructor (translation enabled).

- Parser (istream &, ostream &)

  Constructor (translation disabled).

- void parseDocument ()

  Parses the desired xqML document.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 xqML::Parser::Parser (istream & *association*, istream & *xqMLdoc*, ostream & *output*)

Constructor (translation enabled).

This constructor is used when the xqA spec is available and translation of xqML Symbols to literal identifiers is desired.

Parameters:

association is the xqA spec file

xqMLdoc is the xqML document that has to be parsed

output is the stream where the parser output is to be sent

4.5.2.2 xqML::Parser::Parser (istream & *xqMLdoc*, ostream & *output*)

Constructor (translation disabled).

This constructor is used when the xqA spec is not available or translation of xqML Symbols is not required.

Parameters:

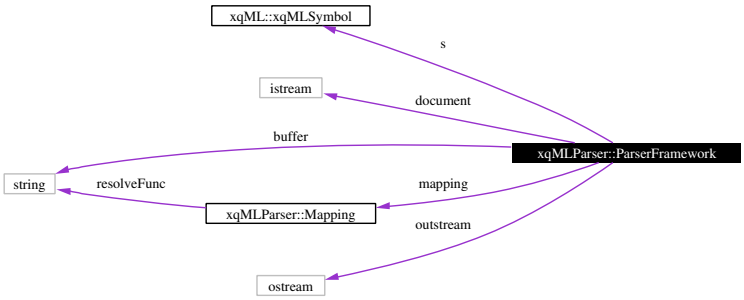
xqMLdoc is the xqML document that has to be parsed

output is the stream where the parser output is to be sent

4.6 xqMLParser::ParserFramework Class Reference

`#include <parser_impl.h>`

Collaboration diagram for xqMLParser::ParserFramework:



4.6.1 Detailed Description

Class to hold private members for Writer.

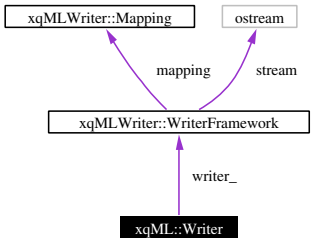
Friends

- class xqML::Parser

4.7 xqML::Writer Class Reference

`#include <xqml/writer.h>`

Collaboration diagram for xqML::Writer:



4.7.1 Detailed Description

Provides API to the client for generation of xqML Documents.

Public Member Functions

- Writer (istream &=std::cin, ostream &=std::cout)

  Constructor takes an istream (default std::cin) for Xqueeze Association Specs and an ostream (default std::cout) for the generated output.

- void attribute (const string &, const string &, bool=false)

  Writes an attribute.

- void cdataSection (const string &)

  Writes a CDATA section that is passed as a string argument.

- void characterReference (int)

  Writes the character reference for the integer argument.

- void comment (const string &)

  Writes a comment.

- void endAttribute ()

  Closes an open attribute.

- void endElement (const string &="")

  Ends an element. The string argument is redundant and defaults to blank.

- void entityReference (const string &)

  Writes an Entity Reference that is passed as the string argument.

- void processingInstruction (const string &, const string &)

  Writes a Processing instruction.

- void prolog (const string &)

  Writes a prolog, declaring the given string argument as doctype.

- void startAttribute (const string &)

  Starts an Attribute.

- void startElement (const string &, bool=false)

  Starts an Element.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 xqML::Writer::Writer (istream & *mapfile* = std::cin, ostream & *outfile* = std::cout) `[explicit]`

Constructor takes an istream (default std::cin) for Xqueeze Association Specs and an ostream (default std::cout) for the generated output.

Constructor requests a `new` WriterFramework to create a Writer object. The default arguments are std::cin and std::cout respectively.

Parameters:

mapfile The xqA specification against which the xqML doc is to be generated

outfile The stream to which the xqML document should be written

4.7.3 Member Function Documentation

4.7.3.1 void xqML::Writer::attribute (const string & *name*, const string & *value*, bool *hasPredefValue* = false)

Writes an attribute.

Write an attribute whose name and value are taken from the first and second arguments respectively. The third argument must be true if the attribute has a predefined value.

Parameters:

name Name of the attribute

value Value of the attribute

hasPredefValue whether the attribute value is known in the xqA

Requires:

Document context should be ”start tag” or ”empty-element start tag”

Throws exception IllegalContext if attribute is not allowed at the point of invocation in the document.

4.7.3.2 void xqML::Writer::cdataSection (const string & *cdata*)

Writes a CDATA section that is passed as a string argument.

Requires:

Document context should be ”element”, ”start tag” or ”empty-element start tag”.

Side Effects:

- Calls xqMLWriter::writerFramework::closePendingElements before writing CDATA.
- Document context is set to ”element” if not already so.

Throws exception IllegalContext if CDATA is not allowed at the point of invocation in the document.

4.7.3.3 void xqML::Writer::characterReference (int *character_code*)

Writes the character reference for the integer argument.

Requires:

Document context should be ”element” or ”open attribute” (redundant).

Side Effects:

Sets Document context to ”element” if earlier it was ”start tag” or ”empty-element start tag”.

Throws exception IllegalContext if character reference is not allowed at the point of invocation in the document.

4.7.3.4 void xqML::Writer::comment (const string & *comment*)

Writes a comment.

Side Effects:

Calls xqMLWriter::writerFramework::closePendingElements before writing the comment.

Throws exception IllegalContext if comment is not allowed at the point of invocation in the document.

4.7.3.5 void xqML::Writer::endAttribute ()

Closes an open attribute.

Requires:
Document context should be ”open attribute”

Side Effects:
Sets context to ”start tag”

Throws exception IllegalContext if closing an attribute is not allowed at the point of invocation in the document or IllegalIdentifier if identifier look-up fails.

4.7.3.6 void xqML::Writer::endElement (const string & *element_to_close* = ””)

Ends an element. The string argument is redundant and defaults to blank.

Requires:
The calls to endElement must match those of startElement and this function should never be called for empty elements.

Guarantees that no more than 255 elements are left pending, by calling xqMLWriter::writer-Framework::closePendingElements when the count reaches 255 before incrementing.

Throws exception IllegalContext if closing an element is not allowed at the point of invocation in the document or IllegalIdentifier if identifier look-up fails.

4.7.3.7 void xqML::Writer::entityReference (const string & *name*)

Writes an Entity Reference that is passed as the string argument.

Requires:
Document context should be ”start tag”, ”empty-element start tag”, ”open attribute” or ”element”.

Side Effects:
Sets Document context to ”element” if not already so.

Throws exception IllegalContext if entity reference is not allowed at the point of invocation in the document or IllegalIdentifier if identifier look-up fails.

4.7.3.8 void xqML::Writer::processingInstruction (const string & *target*, const string & *data*)

Writes a Processing instruction.

Parameters:
target The PI Target

data PI Data

Requires:
context should not be ”open attribute”

Side Effects:
Sets context to ”element” if it is not ”preamble”

4.7.3.9 void xqML::Writer::prolog (const string & *doctype*)

Writes a prolog, declaring the given string argument as doctype.

Writes a prolog, declaring xqml version number (version number should be determined by the function itself). It also declares the name provided as the argument to be the doctype.

4.7.3.10 void xqML::Writer::startAttribute (const string & *name*)

Starts an Attribute.

Starts an attribute whose name is supplied as the argument. This method should not be called for attributes with enumerated values otherwise Writer would throw IllegalIdentifier exceptions.

Requires:
Context should be ”start tag” or ”empty-element start tag”

Side Effects:
Sets context to ”open attribute”

Throws exception IllegalContext if attribute is not allowed at the point of invocation in the document or IllegalIdentifier if identifier look-up fails.

4.7.3.11 void xqML::Writer::startElement (const string & *name*, bool *isEmpty* = false)

Starts an Element.

Starts an element whose name is supplied as the first argument. The optional second argument must be set to `true` if the element is empty. It defaults to `false`.

Requires:
Document context should not be ”open attribute”

Side Effects:

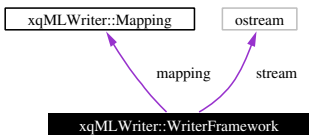
- Calls xqMLWriter::writerFramework::closePendingElements element before writing out anything
- Sets Document context to ”start tag” if the second argument is not supplied or is false, otherwise sets Document context to ”empty element start tag”.

Throws exception IllegalContext if new element is not allowed at the point of invocation in the document or IllegalIdentifier if identifier look-up fails.

4.8 xqMLWriter::WriterFramework Class Reference

`#include <writer_impl.h>`

Collaboration diagram for xqMLWriter::WriterFramework:



4.8.1 Detailed Description

Class to hold private members for Writer.

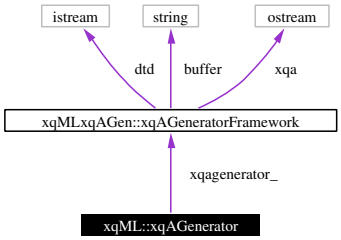
Friends

- class xqML::Writer

4.9 xqML::xqAGenerator Class Reference

`#include <xqml/xqagenerator.h>`

Collaboration diagram for xqML::xqAGenerator:



4.9.1 Detailed Description

This class is used for creating an xqA out of a DTD.

Public Member Functions

- xqAGenerator (istream &dtd, ostream &xqAspecs)

  Constructor.

- void generate (bool preprocess=false)

  Generates the xqA specs for the given DTD.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 xqML::xqAGenerator::xqAGenerator (istream & *dtd*, ostream & *xqAspecs*)

Constructor.

Constructor takes an std::istream reference for the DTD and an std::ostream reference for xqA specification output. The constructor simply allocates a new xqAGeneratorFramework object and passes all it's arguments to the xqAGeneratorFramework constructor.

4.9.3 Member Function Documentation

4.9.3.1 void xqML::xqAGenerator::generate (bool *preprocess* = false)

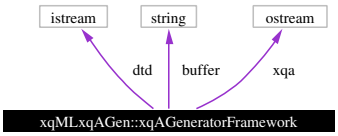
Generates the xqA specs for the given DTD.

Calls xqMLxqAGen::xqAGeneratorFramework::generator() and passes it's argument to it. The parameter "preprocess" should be set to true if the DTD has Parameter Entities (not yet supported). It is false by default.

4.10 xqMLxqAGen::xqAGeneratorFramework Class Reference

`#include <xqagenerator_impl.h>`

Collaboration diagram for xqMLxqAGen::xqAGeneratorFramework:



4.10.1 Detailed Description

Class to hold private members for xqAGenerator.

Friends

- class xqML::xqAGenerator

4.11 xqML::xqMLSymbol Class Reference

`#include <xqml/xqmlsymbol.h>`

4.11.1 Detailed Description

Provides a data type for xqML Symbols and I/O facilities.

Public Member Functions

- xqMLSymbol (unsigned int symbol=0)

  Constructor sets the value of the Symbol to the (optional) argument or zero.

- unsigned int getCode () const

  Returns numeric value of the Symbol.

Friends

- ostream & operator<< (ostream &, xqMLSymbol)

  Overloads operator << for xqML Symbol output.

- istream & operator>> (istream &, xqMLSymbol &)

  Overloads operator >> for xqML Symbol input.

4.11.2 Friends And Related Function Documentation

4.11.2.1 ostream& operator<< (ostream & *stream*, xqMLSymbol *symbol*) [friend]

Overloads operator << for xqML Symbol output.

This operator is capable of writing any unsigned integer in network byte order since it doesn't check whether the xqMLSymbol is indeed valid (i.e. properly uses the lsb as a continuation flag)

4.11.2.2 istream& operator>> (istream & *stream*, xqMLSymbol & *symbol*) [friend]

Overloads operator >> for xqML Symbol input.

Can't handle an xqMLSymbol whose value is greater than the largest value an unsigned integer can hold on the platform used.

5 GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

5.1 Applicability and Definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ”Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ”Transparent” is called ”Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section ”Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ”Acknowledgements”, ”Dedications”, ”Endorsements”, or ”History”.) To ”Preserve the Title” of such a section when you modify the Document means that it remains a section ”Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

5.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

5.3 Copying in Quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

5.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this

License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

5.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

5.7 Aggregation with Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

5.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warrany Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled ”Acknowledgements”, ”Dedications”, or ”History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

5.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ”GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the ”with...Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.