XMLPreProcessor

Generated by Doxygen 1.8.6

Thu Apr 23 2015 16:29:26

Contents

Chapter 1

Main Page

This library is intended to be used to create and load XML files to a C structure.

It is a 2-part software described hereafter.

The first part is a C++ library that is used to parse and retrieve values from XML file, and create XMl files from a list of specifications. This library is described in section C++ Library

The second part is a python script that generates a C code to automatically create XML from a C structure, fill the C structure from an XML file and compare an XML file with the C structure to find the common and and different items. The generated C interface is described in section C Interface The usage of the python script is described in xmlpp.py script

For people only willing to use this library in their code without necessarily understand how it works and what are the additional features provided by the C++ API, reading the sections C Interface and xmlpp.py script is probably sufficient.

1.1 requirements

This library requires the following dependencies to be installed:

- libxml2 and libxlm2-devel
- boost-devel

1.2 C++ Library

This C++ library is a XML parser based on libxml2. As an example, the following structure will be used

```
/*
 * ex_struct.h
 *
 *  Created on: 18 Apr 2015
 *      Author: nlurkin
 */

#ifndef EX_STRUCT_H_
#define EX_STRUCT_H_

#include "xmlstring.h"
#include "hexinteger.h"

typedef struct exampleSubStruct_t {
        int my_integer;
        double my_double;
} exampleSubStruct;

typedef struct exampleStruct_t {
        hexinteger version;
        xmlchar name[XMLSTRING];
        int my_array[5];
        exampleSubStruct my_substruct;
} exampleStruct;

#endif /* EX_STRUCT_H_ */
```

The corresponding complete XML file is

```
<?xml version="1.0"?>
<exampleStruct>
  <version>0x2A</version>
  <name>my_struct</name>
  <my_array id="0">1</my_array>
  <my_array id="1">2</my_array>
  <my_array id="2">3</my_array>
  <my_array id="3">4</my_array>
  <my_array id="4">5</my_array>
  <my_substruct>
    <my_integer>25</my_integer>
    <my_double>36.4</my_double>
  </my_substruct>
</exampleStruct>
```

The following XML is going to be used as an example of partial XML also containing non existing fields

```
<?xml version="1.0"?>
<exampleStruct>
  <name>name from partial</name>
  <external_var>3</external_var>
  <my_substruct>
    <my_double>1024.4</my_double>
  </my_substruct>
</exampleStruct>
```

Both XMLConfParser and XMLConfWriter modules relies on the XMLConfDocument class. The latter provides the back-end for navigating the XML document and will not be described here.

1.2.1 XMLConfParser

It provides a useful API to retrieve values from the XML files through the getValue() method

```
bool getValue(std::string path, int &ref);
bool getValue(std::string path, unsigned int &ref);
bool getValue(std::string path, float &ref);
bool getValue(std::string path, double &ref);
bool getValue(std::string path, char *ref);
bool getValue(std::string path, std::string &ref);
```

The path used in these methods is a string representation of the structure field. Using the example struct (full code can be found in Example/main_parser.cc):

```
        XMLConfParser parser;
        exampleStruct test;
        parser.readFile("partial_ex_struct.xml");
        parser.getValue("exampleStruct.version", test.version);
        parser.getValue("exampleStruct.my_substruct.my_double", test.my_substruct.my_double);
```

It also provides methods to check the content of the XML file. It can tell whether the tag for a specific field of the C structure is found in the XML:

```
        if(!parser.pathExists("exampleStruct.name"))
                cout << "pe: exampleStruct.name is not found in the XML" << endl;
        if(!parser.pathExists("exampleStruct.my_substruct.my_integer"))
                cout << "pe: exampleStruct.my_substruct.my_integer is not found in the XML" << endl;
        if(!parser.pathExists("exampleStruct.my_substruct.my_double"))
                cout << "pe: exampleStruct.my_substruct.my_double is not found in the XML" << endl;
```

Which produces the following output:

```
pe: exampleStruct.my_substruct.my_integer is not found in the XML
```

It can print a list of tags that are present in the XML but not corresponding to any of the paths provided with the addCheckElement() method. This can be used to check for tags present in the XML but not corresponding to any of the field present in the C structure:

```
        parser.startCheckAdditional();
        parser.addCheckElement("exampleStruct.version");
        parser.addCheckElement("exampleStruct.my_array[0]");
        parser.addCheckElement("exampleStruct.my_array[1]");
        cout << "The following fields are present in the XML but not in the above addCheckElement" << endl;
        parser.printAdditional();
```

This piece of code will output:

```
The following fields are present in the XML but not in the above addCheckElement
exampleStruct.external_var
exampleStruct.my_substruct.my_double
exampleStruct.name
```

The last feature offered is the possibility to get the list of paths you are interested in that are present in the XML. This list is created from the paths used on pathExists(). In the example we called pathExists() for

- exampleStruct.name
- exampleStruct.my_substruct.my_integer
- exampleStruct.my_substruct.my_double

  And the xml provides new values for

  - exampleStruct.name
  - exampleStruct.external_var
  - exampleStruct.my_substruct.my_double

  The code hereafter shows the usage of this feature:

  ```
          string diff = parser.getFirstDiff();
          while(diff.compare("")!=0){
                  cout << "d: " << diff << " has received a new value" << endl;
                  diff = parser.getNextDiff();
          }
  ```

  And produces the following output:

  ```
  d: exampleStruct.name has received a new value
  d: exampleStruct.my_substruct.my_double has received a new value
  ```

1.2.2 XMLConfWriter

It provides a useful API to write values in the XML files through the addPath() method

```
bool addPath(std::string path, unsigned int ref);
bool addPath(std::string path, int ref);
bool addPath(std::string path, float ref);
bool addPath(std::string path, double ref);
bool addPath(std::string path, char* ref);
bool addPath(std::string path, std::string ref);
bool addPathAsHex(std::string path, int ref);
```

The path used in these methods is a string representation of the structure field. Using the example struct (full code can be found in Example/main_writer.cc):

We must first create the in-memory XML document and initialise the structure:

```
XMLConfWriter writer;
exampleStruct test;

test.version = 42;
strcpy(test.name, "my_example");
test.my_substruct.my_double = 5.5;

writer.createDocument("exampleStruct");
```

Then the desired elements of the structure are added to the XML document.

```
writer.addPathAsHex("exampleStruct.version", test.version);
writer.addPath("exampleStruct.name", test.name);
writer.addPath("exampleStruct.my_substruct.my_double", test.my_substruct.my_double);
```

And finally the document is written on disk

```
writer.writeDocument("example_writer.xml");
```

The resulting XML file is

```
<?xml version="1.0"?>
<exampleStruct>
  <version>0x2a</version>
  <name>my_example</name>
  <my_substruct>
    <my_double>5.5</my_double>
  </my_substruct>
</exampleStruct>
```

1.3 xmlpp.py script

This python script takes as input a header file describing C structures and generates the code for the automatic structure filling and XML creation. If more than one structure is present in the header file, you must specify which one is the top one. Use it with:

```
xmlpp.py {headerFile.h} --struct {topStructure}
```

This will produce besides your {headerFile.h} two C++ files {headerFileProxy.h} and {headerFileProxy.cc} that you should include in your compilation.

1.4 C Interface

This C interface is probably the only part that most users will use. Starting from the example structure given above, the xmlpp.py script will generate ex_structProxy.h and ex_structProxy.cc. These files provide the following functions, callable from C code:

```
int xml_read_file_exampleStruct(const char* fileName);
int xml_apply_exampleStruct(exampleStruct *ptr);
int xml_test_exampleStruct();
void* xml_start_compare_exampleStruct(exampleStruct *ptr);
void* xml_next_compare_exampleStruct(exampleStruct *ptr);
int xml_create_exampleStruct(exampleStruct *ptr, const char* fileName);
const char* xml_getLastFatalError_exampleStruct();
```

- `int xml_read_file_exampleStruct(const char* fileName);`

  Use this function to read a new XML file.

  fileName is the full path to the XML file to read.

- `int xml_apply_exampleStruct(exampleStruct *ptr);`

  Use this function to apply the loaded XML file to your structure.

  ptr is a pointer to the structure you want to fill.

- `int xml_test_exampleStruct();`

  Use this function to print the list of elements in the structure that are not present in the XML and print the list of tags present in the XML that are not in the structure.

- `void* xml_start_compare_exampleStruct(exampleStruct *ptr);`

  The next two functions work together and are used to get the list of fields that received a new value from the XML. This one should be used first and returns a pointer to the first modified element. If no element has been modified it return a NULL pointer.

- `void* xml_next_compare_exampleStruct(exampleStruct *ptr);`

  This function will return a pointer to the next modified element. After reaching the last element this function will return a NULL pointer.

- `int xml_create_exampleStruct(exampleStruct *ptr, const char* fileName);`

  Use this function to create an XML file corresponding to the structure. The current values of the structure are set in the file.

  ptr is a pointer to the reference structure. fileName is the full path to the XML file to write.

- `const char* xml_getLastFatalError_exampleStruct();`

  If one of the function returns with an error code (-1), use this function to retrieve the last fatal error that occurred.

  The fatal error will warn you about malformed XML files, specifying what is the error and where it can be found.
  An example of error occurring when an end tag has a typo:

  ```
  Fatal error: Document not parsed successfully.
  File: partial.xml at (l:5, c:27): expected '>
  ```

  Or for an opening tag without its closing tag:

  ```
  Fatal error: Document not parsed successfully.
  File: partial.xml at (l:8, c:1): Premature end of data in tag my_struct line 2
  ```

A fully working example using this interface can be found in Example/main_proxy.cc

Start by declaring the structure and reading the new XML file:

```
exampleStruct test;

if(xml_read_file_exampleStruct("ex_struct.xml")==-1)
        cout << "Fatal error: " << xml_getLastFatalError_exampleStruct() << endl;
```

Then apply the content of the file to structure to initialise it:

```
xml_apply_exampleStruct(&test);
```

We can then do the operation again with a partial XML file

```
if(xml_read_file_exampleStruct("partial_ex_struct.xml")==-1)
        cout << "Fatal error: " << xml_getLastFatalError_exampleStruct() << endl;
xml_apply_exampleStruct(&test);
```

The structure now contains the following values:

```
val: test.version= 42
val: test.name= name from partial
val: test.my_array[0]= 1
val: test.my_array[1]= 2
val: test.my_array[2]= 3
val: test.my_array[3]= 4
val: test.my_array[4]= 5
val: test.my_substruct.my_double= 1024.4
val: test.my_substruct.my_integer= 25
```

Then we print the list of differences between the structure and the file

```
        xml_test_exampleStruct();
```

giving the following output:

```
Differences:
exampleStruct.version was not found in XML file
exampleStruct.my_array[0] was not found in XML file
exampleStruct.my_array[1] was not found in XML file
exampleStruct.my_array[2] was not found in XML file
exampleStruct.my_array[3] was not found in XML file
exampleStruct.my_array[4] was not found in XML file
exampleStruct.my_substruct.my_integer was not found in XML file
XML tags without struct correspondance:
exampleStruct.external_var
```

We can determine which fields of the structure have been modified in the XML. We request the pointer to the first modified element and we can compare the address of the pointer with the address of our structure fields to determine which one. We then loop on the next elements until the pointer becomes NULL, indicating the past-the-end element.

```
        void* ptr = xml_start_compare_exampleStruct(&test);

        while(ptr!=NULL){
                if(ptr==&(test.version)) cout << "mod: test.version was modified" << endl;
                else if(ptr==&(test.name)) cout << "mod: test.name was modified" << endl;
                else if(ptr==&(test.my_substruct.my_double)) cout << "mod: test.my_substruct.my_double was
       modified" << endl;
                else if(ptr==&(test.my_substruct.my_integer)) cout << "mod: test.my_substruct.my_integer
       was modified" << endl;
                else if(ptr==&(test.my_array[2])) cout << "mod: test.my_array[2] was modified" << endl;
                ptr = xml_next_compare_exampleStruct(&test);
        }
```

This outputs the following:

```
mod: test.name was modified
mod: test.my_substruct.my_double was modified
```

We can finally finish by modifying the structure and write the final XML:

```
        strcpy(test.name, "my_modified_struct");
        xml_create_exampleStruct(&test, "outputStruct.xml");
```

Which is

```
<?xml version="1.0"?>
<exampleStruct>
  <version>0x2a</version>
  <name>my_modified_struct</name>
  <my_array id="0">1</my_array>
  <my_array id="1">2</my_array>
  <my_array id="2">3</my_array>
  <my_array id="3">4</my_array>
  <my_array id="4">5</my_array>
  <my_substruct>
    <my_integer>25</my_integer>
    <my_double>1024.4</my_double>
  </my_substruct>
</exampleStruct>
```

Chapter 2

list_pe

- exampleStruct.name
- exampleStruct.my_substruct.my_integer
- exampleStruct.my_substruct.my_double

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 bumpversion Namespace Reference

7.2 xmlpp Namespace Reference

Functions

- def parseArgs
- def prepareProxy
- def preProcessFile
- def printHelp
- def test

Variables

- tuple __descr__ = ("""Script for XML proxy""")
- string __version__ = "1.0.0"
- list filePath = sys.argv[1]
- string topStruct = ""

7.2.1 Function Documentation

7.2.1.1 def xmlpp.parseArgs ( )

Definition at line 102 of file xmlpp.py.

7.2.1.2 def xmlpp.prepareProxy (*args*)

```
Create the proxy functions for the structure given in the header file
```

Definition at line 43 of file xmlpp.py.

7.2.1.3 def xmlpp.preProcessFile (*filePath*)

Definition at line 21 of file xmlpp.py.

7.2.1.4 def xmlpp.printHelp (*args*)

Definition at line 99 of file xmlpp.py.

7.2.1.5 def xmlpp.test (*args*)

```
Test the parsing of the header file.
It should print something equivalent to the input header file
```

Definition at line 73 of file xmlpp.py.

7.2.2 Variable Documentation

7.2.2.1 tuple xmlpp.__descr__ = ("""Script for XML proxy""")

Definition at line 17 of file xmlpp.py.

7.2.2.2 string xmlpp.__version__ = "1.0.0"

Definition at line 16 of file xmlpp.py.

7.2.2.3 list xmlpp.filePath = sys.argv[1]

Definition at line 135 of file xmlpp.py.

7.2.2.4 list xmlpp.topStruct = ""

Definition at line 125 of file xmlpp.py.

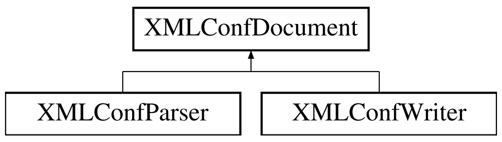
Chapter 8

Class Documentation

8.1 XMLConfDocument Class Reference

```
#include <XMLConfDocument.h>
```

Inheritance diagram for XMLConfDocument:



Public Member Functions

- void closeFile ()

  Close the current document. Free allocated memory.
- xmlNodePtr findArrayNode (std::string nodeName, int index, xmlNodePtr node)

  Find a child of node with name nodeName and array index index.
- xmlNodePtr findChildNode (std::string nodeName, xmlNodePtr node)

  Find a child of node with name nodeName.
- xmlNodePtr findNextSiblingNode (std::string nodeName, xmlNodePtr node)

  Find next sibling of node with name nodeName.
- xmlNodePtr findPathNode (std::string path)

  Find a node corresponding to path.
- xmlNodePtr findPathNode (std::vector$<$ std::string $>$ path, xmlNodePtr cur)

  Find a node corresponding to path under the current node.
- XMLErrorStack getLastError ()

  Return the error stack.
- std::string getNodeString (xmlNodePtr node)

  Return a the value of the node as a string.
- int getNSiblings (xmlNodePtr node)

  Return the number of siblings of node. node is included in the count.
- xmlNodePtr getRoot () const

  Return the root node of the document.
- int isArrayNode (std::string &name)

  Does the provided name corresponds to an array element?
- int isArrayNode (xmlNodePtr node)

Does the provided node corresponds to an array element?
- bool isDocumentInitialised ()

  Is the XML document initialised.
- void printNode (xmlNodePtr node)

  Print the node.
- void printNodeValue (xmlNodePtr node)

  Print the value of node.
- std::string readAttribute (std::string attributeName, xmlNodePtr node)

  Read an attribute of the node.
- XMLConfDocument ()

  Constructor.
- virtual ∼XMLConfDocument ()

  Destructor.

Protected Attributes

- xmlDocPtr fDoc

  Pointer to the XML document.
- XMLErrorStack fErrorStack

  Error stack.
- xmlNodePtr fRoot

  Pointer to the root node of the document.

8.1.1 Detailed Description

Back-end class to navigate the XML file.

Definition at line 55 of file XMLConfDocument.h.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 XMLConfDocument::XMLConfDocument ( ) `[inline]`

Constructor.

Definition at line 58 of file XMLConfDocument.h.

8.1.2.2 virtual XMLConfDocument::∼XMLConfDocument ( ) `[inline]`, `[virtual]`

Destructor.

Definition at line 60 of file XMLConfDocument.h.

8.1.3 Member Function Documentation

8.1.3.1 void XMLConfDocument::closeFile ( )

Close the current document. Free allocated memory.

Definition at line 70 of file XMLConfDocument.cpp.

8.1.3.2 xmlNodePtr XMLConfDocument::findArrayNode (std::string *nodeName*, int *index*, xmlNodePtr *node*)

Find a child of node with name nodeName and array index index.

Parameters

nodeName	Searched child array name(tag)
index	Searched array index
node	Pointer to a node

Returns

If found, pointer to the node. Else NULL pointer.

Definition at line 136 of file XMLConfDocument.cpp.

8.1.3.3 xmlNodePtr XMLConfDocument::findChildNode (std::string *nodeName,* xmlNodePtr *node*)

Find a child of node with name nodeName.

Parameters

nodeName	Searched child name(tag)
node	Pointer to a node

Returns

If found, pointer to the node. Else NULL pointer.

Definition at line 38 of file XMLConfDocument.cpp.

8.1.3.4 xmlNodePtr XMLConfDocument::findNextSiblingNode (std::string *nodeName,* xmlNodePtr *node*)

Find next sibling of node with name nodeName.

Parameters

nodeName	Searched sibling name(tag)
node	Pointer to a node

Returns

If found, pointer to the node. Else NULL pointer.

Definition at line 53 of file XMLConfDocument.cpp.

8.1.3.5 xmlNodePtr XMLConfDocument::findPathNode (std::string *path*)

Find a node corresponding to path.

Warning

Fills the error stack

Parameters

path	Path a.b.c corresponds to the XML structure **1**

Returns

If found, pointer to the node. Else NULL pointer.

Definition at line 97 of file XMLConfDocument.cpp.

8.1.3.6 xmlNodePtr XMLConfDocument::findPathNode (std::vector< std::string > *path*, xmlNodePtr *cur*)

Find a node corresponding to path under the current node.

Warning

Fills the error stack

Parameters

path	Vectorised path a.b.c is vectorised as {a,b,c} and corresponds to the XML structure **1**
cur	Pointer to a node

Returns

If found, pointer to the node. Else NULL pointer.

Definition at line 173 of file XMLConfDocument.cpp.

8.1.3.7 XMLErrorStack XMLConfDocument::getLastError ( ) `[inline]`

Return the error stack.

Returns

Error stack

Definition at line 92 of file XMLConfDocument.h.

8.1.3.8 std::string XMLConfDocument::getNodeString (xmlNodePtr *node*)

Return a the value of the node as a string.

Parameters

node	Pointer to a node

Returns

String representation of the node value if found. Else empty string.

Definition at line 82 of file XMLConfDocument.cpp.

8.1.3.9 int XMLConfDocument::getNSiblings (xmlNodePtr *node*)

Return the number of siblings of node. node is included in the count.

Parameters

node	Pointer to a node

Returns

Number of siblings of the provided node. If node is not NULL, the returned value is greater than 0.

Definition at line 229 of file XMLConfDocument.cpp.

8.1.3.10 xmlNodePtr XMLConfDocument::getRoot ( ) const [inline]

Return the root node of the document.

Returns

Pointer to the root node.

Definition at line 85 of file XMLConfDocument.h.

8.1.3.11 int XMLConfDocument::isArrayNode (std::string & *name*)

Does the provided name corresponds to an array element?

Parameters

name	name to check. my_name[12] corresponds to an array.

Returns

Index of the array or -1 if not an array.

Definition at line 157 of file XMLConfDocument.cpp.

8.1.3.12 int XMLConfDocument::isArrayNode (xmlNodePtr *node*)

Does the provided node corresponds to an array element?

<my_tag id="3"></my_tag> is an array node of index 3.

Parameters

node	A pointer to a node. A node with an id attribute compatible with an integer

Returns

Index of the array or -1 if not an array.

Definition at line 243 of file XMLConfDocument.cpp.

8.1.3.13 bool XMLConfDocument::isDocumentInitialised ( ) [inline]

Is the XML document initialised.

Returns

true if XML document is initialised and operations are allowed, else false.

Definition at line 100 of file XMLConfDocument.h.

8.1.3.14 void XMLConfDocument::printNode (xmlNodePtr *node*)

Print the node.

Parameters

node	Pointer to a node

Definition at line 120 of file XMLConfDocument.cpp.

8.1.3.15 void XMLConfDocument::printNodeValue (xmlNodePtr *node*)

Print the value of node.

Parameters

node	Pointer to a node

Definition at line 66 of file XMLConfDocument.cpp.

8.1.3.16 std::string XMLConfDocument::readAttribute (std::string *attributeName,* xmlNodePtr *node*)

Read an attribute of the node.

Parameters

attributeName	Name of the attribute
node	Pointer to a node

Returns

String representation of the attribute if found, or empty string

Definition at line 197 of file XMLConfDocument.cpp.

8.1.4 Member Data Documentation

8.1.4.1 xmlDocPtr XMLConfDocument::fDoc `[protected]`

Pointer to the XML document.

Definition at line 100 of file XMLConfDocument.h.

8.1.4.2 XMLErrorStack XMLConfDocument::fErrorStack `[protected]`

Error stack.

Definition at line 106 of file XMLConfDocument.h.

8.1.4.3 xmlNodePtr XMLConfDocument::fRoot `[protected]`

Pointer to the root node of the document.

Definition at line 104 of file XMLConfDocument.h.

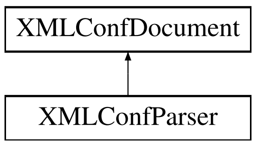
The documentation for this class was generated from the following files:

- XMLConf/XMLConfDocument.h
- XMLConf/XMLConfDocument.cpp

8.2 XMLConfParser Class Reference

`#include <XMLConfParser.h>`

Inheritance diagram for XMLConfParser:



Public Member Functions

- void addCheckElement (std::string path)

  Add an element to the list of provided elements for the check of additional tags.
- void addListDiffElement (std::string path)

  Add an element to the list of modified elements.
- std::string getFirstDiff ()

  Return the first element in the list of modified elements.
- std::string getNextDiff ()

  Return the next element in the list of modified elements.
- int getReadSuccess ()

  Get number of ReadSuccess.
- bool getValue (std::string path, int &ref)

  Get a value as int.
- bool getValue (std::string path, unsigned int &ref)

  Get a value as unsigned int.
- bool getValue (std::string path, float &ref)

  Get a value as float.
- bool getValue (std::string path, double &ref)

  Get a value as double.
- bool getValue (std::string path, char ∗ref)

  Get a value as c-string (do not require conversion)
- bool getValue (std::string path, std::string &ref)

  Get a value as std::string (do not require conversion)
- bool pathExists (std::string path)

  Does the given path exists in the XML file.
- void printAdditional ()

  Print the additional tags.
- bool readFile (std::string fileName)

  Read an XML file.
- void resetReadSuccess ()

  Reset number of ReadSuccess.
- void startCheckAdditional ()

  Initialise the check of additional tags.
- XMLConfParser ()

  Constructor.
- ∼XMLConfParser ()

  Destructor.

Private Member Functions

- void walkTreeCompare (std::string prefix, xmlNodePtr node)

    Recursively walk through the XML tree.

Private Attributes

- std::set< std::string > fListAdditional

    List containing the additional tags.
- std::vector< std::string > fListDiff

    List containing the modified elements.
- std::vector< std::string >
  ::iterator fListDiffIterator

    Iterator through the list containing the modified elements.
- int fReadSuccess

    Counter for the number of successful reads.

Additional Inherited Members

8.2.1 Detailed Description

Class for reading and parsing XML files.

Definition at line 28 of file XMLConfParser.h.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 XMLConfParser::XMLConfParser ( ) [inline]

Constructor.

Definition at line 31 of file XMLConfParser.h.

8.2.2.2 XMLConfParser::∼XMLConfParser ( ) [inline]

Destructor.

Definition at line 33 of file XMLConfParser.h.

8.2.3 Member Function Documentation

8.2.3.1 void XMLConfParser::addCheckElement (std::string *path*)

Add an element to the list of provided elements for the check of additional tags.

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>

Definition at line 269 of file XMLConfParser.cpp.

8.2.3.2 void XMLConfParser::addListDiffElement (std::string *path*)

Add an element to the list of modified elements.

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>

Definition at line 285 of file XMLConfParser.cpp.

8.2.3.3 std::string XMLConfParser::getFirstDiff ( )

Return the first element in the list of modified elements.

Returns

Path of the first element in the list of modified elements. Empty string if the list is empty.

Definition at line 223 of file XMLConfParser.cpp.

8.2.3.4 std::string XMLConfParser::getNextDiff ( )

Return the next element in the list of modified elements.

Returns

Path of the next element in the list of modified elements. Empty string if the iteration through the list is over.

Definition at line 233 of file XMLConfParser.cpp.

8.2.3.5 int XMLConfParser::getReadSuccess ( ) `[inline]`

Get number of ReadSuccess.

ReadSuccess is a counter that is incremented for every successful getValue() or pathExists().

A getValue is successful if the requested node exists and the contained value can be successfully read and transformed into the requested type.

A pathExists is successful if the path is found in the XML.

Returns

Number of ReadSuccess

Definition at line 46 of file XMLConfParser.h.

8.2.3.6 bool XMLConfParser::getValue (std::string *path*, int & *ref*)

Get a value as int.

Fill the variable passed by reference with the value found at the specified path in the XML. If the path does not exist in the XML, the variable is untouched.

Warning

Fill the error stack

Parameters

path	Path to retrieve. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Variable to fill the the value found at path.

Returns

true in case of success (path is found and the value can be transformed into int). Else false.

Definition at line 51 of file XMLConfParser.cpp.

8.2.3.7 bool XMLConfParser::getValue (std::string *path,* unsigned int & *ref*)

Get a value as unsigned int.

Fill the variable passed by reference with the value found at the specified path in the XML. If the path does not exist in the XML, the variable is untouched.

Warning

Fill the error stack

Parameters

path	Path to retrieve. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Variable to fill the the value found at path.

Returns

true in case of success (path is found and the value can be transformed into unsigned int). Else false.

Definition at line 79 of file XMLConfParser.cpp.

8.2.3.8 bool XMLConfParser::getValue (std::string *path,* float & *ref*)

Get a value as float.

Fill the variable passed by reference with the value found at the specified path in the XML. If the path does not exist in the XML, the variable is untouched.

Warning

Fill the error stack

Parameters

path	Path to retrieve. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Variable to fill the the value found at path.

Returns

true in case of success (path is found and the value can be transformed into float). Else false.

Definition at line 107 of file XMLConfParser.cpp.

8.2.3.9 bool XMLConfParser::getValue (std::string *path,* double & *ref*)

Get a value as double.

Fill the variable passed by reference with the value found at the specified path in the XML. If the path does not exist in the XML, the variable is untouched.

Warning

Fill the error stack

Parameters

path	Path to retrieve. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Variable to fill the the value found at path.

Returns

true in case of success (path is found and the value can be transformed into double). Else false.

Definition at line 135 of file XMLConfParser.cpp.

8.2.3.10 bool XMLConfParser::getValue (std::string *path,* char ∗ *ref*)

Get a value as c-string (do not require conversion)

Fill the variable passed by reference with the value found at the specified path in the XML. If the path does not exist in the XML, the variable is untouched.

Parameters

path	Path to retrieve. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Variable to fill the the value found at path.

Returns

true in case of success (path is found). Else false.

Definition at line 180 of file XMLConfParser.cpp.

8.2.3.11 bool XMLConfParser::getValue (std::string *path,* std::string & *ref*)

Get a value as std::string (do not require conversion)

Fill the variable passed by reference with the value found at the specified path in the XML. If the path does not exist in the XML, the variable is untouched.

Parameters

path	Path to retrieve. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Variable to fill the the value found at path.

Returns

true in case of success (path is found). Else false.

Definition at line 162 of file XMLConfParser.cpp.

8.2.3.12 bool XMLConfParser::pathExists (std::string *path*)

Does the given path exists in the XML file.

Parameters

path	Path whose existence is checked. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>

Returns

true if the path is found in the XML, else false.

Definition at line 196 of file XMLConfParser.cpp.

8.2.3.13 void XMLConfParser::printAdditional ( )

Print the additional tags.

Definition at line 273 of file XMLConfParser.cpp.

8.2.3.14 bool XMLConfParser::readFile (std::string *fileName*)

Read an XML file.

Parameters

fileName	Full path to the XML file to be read

Returns

true if the file could be read and parsed. Else false.

Definition at line 18 of file XMLConfParser.cpp.

8.2.3.15 void XMLConfParser::resetReadSuccess ( ) `[inline]`

Reset number of ReadSuccess.

Definition at line 48 of file XMLConfParser.h.

8.2.3.16 void XMLConfParser::startCheckAdditional ( )

Initialise the check of additional tags.

The check for additional tags prints a list of tags present in the XML file but not present in the list of provided elements.

Definition at line 211 of file XMLConfParser.cpp.

8.2.3.17 void XMLConfParser::walkTreeCompare (std::string *prefix,* xmlNodePtr *node*) `[private]`

Recursively walk through the XML tree.

Recursively find all the paths found in the XML tree and add them in the list of modified elements.

Parameters

prefix	Current prefix for the path

node	Pointer to a node

Definition at line 244 of file XMLConfParser.cpp.

8.2.4 Member Data Documentation

8.2.4.1 std::set<std::string> XMLConfParser::fListAdditional [private]

List containing the additional tags.

Definition at line 69 of file XMLConfParser.h.

8.2.4.2 std::vector<std::string> XMLConfParser::fListDiff [private]

List containing the modified elements.

Definition at line 70 of file XMLConfParser.h.

8.2.4.3 std::vector<std::string>::iterator XMLConfParser::fListDiffIterator [private]

Iterator through the list containing the modified elements.

Definition at line 71 of file XMLConfParser.h.

8.2.4.4 int XMLConfParser::fReadSuccess [private]

Counter for the number of successful reads.

Definition at line 68 of file XMLConfParser.h.

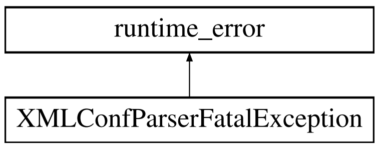
The documentation for this class was generated from the following files:

- XMLConf/XMLConfParser.h
- XMLConf/XMLConfParser.cpp

8.3 XMLConfParserFatalException Class Reference

`#include <XMLConfParser.h>`

Inheritance diagram for XMLConfParserFatalException:



Public Member Functions

- XMLConfParserFatalException (std::string message)
- XMLConfParserFatalException (const std::stringstream &message)

8.3.1 Detailed Description

Definition at line 17 of file XMLConfParser.h.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 XMLConfParserFatalException::XMLConfParserFatalException (std::string *message*) `[inline]`

Definition at line 20 of file XMLConfParser.h.

8.3.2.2 XMLConfParserFatalException::XMLConfParserFatalException (const std::stringstream & *message*) `[inline]`

Definition at line 21 of file XMLConfParser.h.

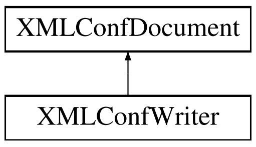
The documentation for this class was generated from the following file:

- XMLConf/XMLConfParser.h

8.4 XMLConfWriter Class Reference

`#include <XMLConfWriter.h>`

Inheritance diagram for XMLConfWriter:



Public Member Functions

- bool addPath (std::string path, int ref)

  Add a new path to the document.
- bool addPath (std::string path, unsigned int ref)

  Add a new path to the document.
- bool addPath (std::string path, float ref)

  Add a new path to the document.
- bool addPath (std::string path, double ref)

  Add a new path to the document.
- bool addPath (std::string path, char ∗ref)

  Add a new path to the document.
- bool addPath (std::string path, std::string ref)

  Add a new path to the document.
- bool addPathAsHex (std::string path, int ref)

  Add a new path to the document as hexadecimal integer.
- void createDocument (std::string structName)

  Create a new XML document with the specified root node.
- void printDocument ()

  Print the XML document in the output.
- bool writeDocument (std::string fileName)

  Write the document at the specified path.
- XMLConfWriter ()

  Constructor.
- virtual ∼XMLConfWriter ()

  Destructor.

Private Member Functions

- xmlNodePtr addNode (std::string nodeName, xmlNodePtr node)

  Add a new child node.
- xmlNodePtr addNodeArray (std::string nodeName, int index, xmlNodePtr node)

  Add a new child node array.
- void addNodeValue (std::string value, xmlNodePtr node)

  Add a value to the node.
- xmlNodePtr addPathNode (std::string path)

  Add a path (string version)
- xmlNodePtr addPathNode (std::vector< std::string > path, xmlNodePtr cur)

  Add a path (vector version)

Additional Inherited Members

8.4.1 Detailed Description

Class for creating XML files

Definition at line 20 of file XMLConfWriter.h.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 XMLConfWriter::XMLConfWriter ( ) `[inline]`

Constructor.

Definition at line 23 of file XMLConfWriter.h.

8.4.2.2 virtual XMLConfWriter::∼XMLConfWriter ( ) `[inline]`, `[virtual]`

Destructor.

Definition at line 25 of file XMLConfWriter.h.

8.4.3 Member Function Documentation

8.4.3.1 xmlNodePtr XMLConfWriter::addNode (std::string *nodeName*, xmlNodePtr *node*) `[private]`

Add a new child node.

Parameters

nodeName	Name of the child node to add
node	Pointer to the parent node

Returns

Pointer to the newly created node

Definition at line 32 of file XMLConfWriter.cpp.

8.4.3.2 xmlNodePtr XMLConfWriter::addNodeArray (std::string *nodeName*, int *index*, xmlNodePtr *node*) `[private]`

Add a new child node array.

An array node as an id attribute giving the index in the array.

Parameters

nodeName	Name of the child node to add
index	Array index of the node
node	Pointer to the parent node

Returns

Pointer to the newly created node

Definition at line 43 of file XMLConfWriter.cpp.

8.4.3.3 void XMLConfWriter::addNodeValue (std::string *value,* xmlNodePtr *node*) [private]

Add a value to the node.

Parameters

value	Value to assign to the node
node	Pointer to a node

Definition at line 57 of file XMLConfWriter.cpp.

8.4.3.4 bool XMLConfWriter::addPath (std::string *path,* int *ref*)

Add a new path to the document.

Add a new path to the XML document, with the value contained in ref

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Reference value

Returns

true if the path could be added, else false

Definition at line 142 of file XMLConfWriter.cpp.

8.4.3.5 bool XMLConfWriter::addPath (std::string *path,* unsigned int *ref*)

Add a new path to the document.

Add a new path to the XML document, with the value contained in ref

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Reference value

Returns

true if the path could be added, else false

Definition at line 125 of file XMLConfWriter.cpp.

8.4.3.6 bool XMLConfWriter::addPath (std::string *path,* float *ref*)

Add a new path to the document.

Add a new path to the XML document, with the value contained in ref

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Reference value

Returns

true if the path could be added, else false

Definition at line 159 of file XMLConfWriter.cpp.

8.4.3.7 bool XMLConfWriter::addPath (std::string *path,* double *ref*)

Add a new path to the document.

Add a new path to the XML document, with the value contained in ref

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Reference value

Returns

true if the path could be added, else false

Definition at line 176 of file XMLConfWriter.cpp.

8.4.3.8 bool XMLConfWriter::addPath (std::string *path,* char $*$ *ref*)

Add a new path to the document.

Add a new path to the XML document, with the value contained in ref

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Reference value

Returns

true if the path could be added, else false

Definition at line 193 of file XMLConfWriter.cpp.

8.4.3.9 bool XMLConfWriter::addPath (std::string *path,* std::string *ref*)

Add a new path to the document.

Add a new path to the XML document, with the value contained in ref

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Reference value

Returns

true if the path could be added, else false

Definition at line 208 of file XMLConfWriter.cpp.

8.4.3.10 bool XMLConfWriter::addPathAsHex (std::string *path,* int *ref*)

Add a new path to the document as hexadecimal integer.

Add a new path to the XML document, with the value contained in ref. The integer value is printed in hexadecimal format (0x..)

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>
ref	Reference value

Returns

true if the path could be added, else false

Definition at line 224 of file XMLConfWriter.cpp.

8.4.3.11 xmlNodePtr XMLConfWriter::addPathNode (std::string *path*) `[private]`

Add a path (string version)

Parameters

path	Path to add. The path "a.b.c" corresponds to the xml structure <a><b><c>1</c></b></a>

Returns

Pointer to the newly created node. NULL pointer if unsuccessful.

Definition at line 67 of file XMLConfWriter.cpp.

8.4.3.12 xmlNodePtr XMLConfWriter::addPathNode (std::vector< std::string > *path,* xmlNodePtr *node*) `[private]`

Add a path (vector version)

Parameters

path	Vectorised path to add to the node. The vectorised path of "a.b.c" is {a,b,c} and corresponds to the xml structure <a><b><c>1</c></b></a>

node	Pointer to a node

Returns

Pointer to the newly created node. NULL pointer if unsuccessful.

Definition at line 87 of file XMLConfWriter.cpp.

8.4.3.13 void XMLConfWriter::createDocument (std::string *structName*)

Create a new XML document with the specified root node.

Parameters

structName	Name of the root node

Definition at line 16 of file XMLConfWriter.cpp.

8.4.3.14 void XMLConfWriter::printDocument ( )

Print the XML document in the output.

Definition at line 103 of file XMLConfWriter.cpp.

8.4.3.15 bool XMLConfWriter::writeDocument (std::string *fileName*)

Write the document at the specified path.

Parameters

fileName	Full path to the file to write

Returns

true if the file has been successfully written, else false.

Definition at line 115 of file XMLConfWriter.cpp.

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

- XMLConf/XMLConfWriter.h
- XMLConf/XMLConfWriter.cpp

8.5 XMLErrorStack Class Reference

```
#include <XMLConfDocument.h>
```

Public Member Functions

- void addError (std::string s)

  Add an error in the stack.
- void addError (std::stringstream &s)

  Add an error in the stack.
- void clear ()

  Clear the stack. Remove all errors.

- void printStack ()

  Print the stack.
- std::string stringStack ()

  Return the stack as a string.
- XMLErrorStack ()

  Constructor.
- virtual ∼XMLErrorStack ()

  Destructor.

Private Attributes

- std::vector< std::string > fStack

  Vector containing the errors.

8.5.1 Detailed Description

Class containing an error stack. This is used to allow printing the error whenever convenient and not when the error happens.

Definition at line 32 of file XMLConfDocument.h.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 XMLErrorStack::XMLErrorStack ( ) [inline]

Constructor.

Definition at line 35 of file XMLConfDocument.h.

8.5.2.2 virtual XMLErrorStack::∼XMLErrorStack ( ) [inline], [virtual]

Destructor.

Definition at line 37 of file XMLConfDocument.h.

8.5.3 Member Function Documentation

8.5.3.1 void XMLErrorStack::addError (std::string *s*) [inline]

Add an error in the stack.

Definition at line 42 of file XMLConfDocument.h.

8.5.3.2 void XMLErrorStack::addError (std::stringstream & *s*) [inline]

Add an error in the stack.

Definition at line 44 of file XMLConfDocument.h.

8.5.3.3 void XMLErrorStack::clear ( ) [inline]

Clear the stack. Remove all errors.

Definition at line 40 of file XMLConfDocument.h.

8.5.3.4 void XMLErrorStack::printStack ( )

Print the stack.

Definition at line 207 of file XMLConfDocument.cpp.

8.5.3.5 std::string XMLErrorStack::stringStack ( )

Return the stack as a string.

Returns

Single string containing all the error stack.

Definition at line 215 of file XMLConfDocument.cpp.

8.5.4 Member Data Documentation

8.5.4.1 std::vector<std::string> XMLErrorStack::fStack `[private]`

Vector containing the errors.

Definition at line 49 of file XMLConfDocument.h.

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

- XMLConf/XMLConfDocument.h
- XMLConf/XMLConfDocument.cpp

Chapter 9

File Documentation

9.1 bumpversion.py File Reference

Namespaces

- bumpversion

9.2 Example/DocGen/parser_list_pe.dox File Reference

9.3 Example/main_parser.cc File Reference

```
#include "XMLConfParser.h"
#include "ex_struct.h"
#include <iostream>
#include <string>
```

Functions

- int main ()

9.3.1 Function Documentation

9.3.1.1 int main ( )

Definition at line 14 of file main_parser.cc.

9.4 Example/main_proxy.cc File Reference

```
#include "ex_struct.h"
#include "ex_structProxy.h"
#include <iostream>
#include <cstring>
```

Macros

- #define PRINTVAR(v) #v << "= " << v << " "

Functions

- int main ()

9.4.1 Macro Definition Documentation

9.4.1.1 #define PRINTVAR(*v*) #v << "= " << v << " "

Definition at line 14 of file main_proxy.cc.

9.4.2 Function Documentation

9.4.2.1 int main ( )

Definition at line 16 of file main_proxy.cc.

9.5 Example/main_writer.cc File Reference

```
#include "XMLConfWriter.h"
#include "ex_struct.h"
#include <iostream>
#include <cstring>
```

Functions

- int main ()

9.5.1 Function Documentation

9.5.1.1 int main ( )

Definition at line 15 of file main_writer.cc.

9.6 main.dox File Reference

9.7 XMLConf/hexinteger.h File Reference

Typedefs

- typedef int hexinteger

9.7.1 Typedef Documentation

9.7.1.1 typedef int **hexinteger**

Definition at line 4 of file hexinteger.h.

9.8 XMLConf/XMLConfDocument.cpp File Reference

```
#include "XMLConfDocument.h"
#include <iostream>
#include <boost/tokenizer.hpp>
```

Functions

- std::vector< std::string > tokenize (std::string s, char const *separator)

  Tokenize a string according to separator.

9.8.1 Function Documentation

9.8.1.1 std::vector<std::string> tokenize (std::string *s*, char const * *separator*)

Tokenize a string according to separator.

Parameters

s	String to tokenize
separator	List of separators

Returns

vector of string containing the tokens

Definition at line 20 of file XMLConfDocument.cpp.

9.9 XMLConf/XMLConfDocument.h File Reference

```
#include <libxml/xmlmemory.h>
#include <libxml/parser.h>
#include <vector>
#include <string>
#include <sstream>
#include "XMLConfVersion.h"
```

Classes

- class XMLConfDocument
- class XMLErrorStack

Functions

- std::vector< std::string > tokenize (std::string s, char const ∗separator)

  Tokenize a string according to separator.
- std::vector< std::string > tokenizePath (std::string s)

  Tokenize a path (separator is .)

9.9.1 Function Documentation

9.9.1.1 std::vector<std::string> tokenize (std::string *s*, char const ∗ *separator*)

Tokenize a string according to separator.

Parameters

s	String to tokenize
separator	List of separators

Returns

vector of string containing the tokens

Definition at line 20 of file XMLConfDocument.cpp.

9.9.1.2 std::vector<std::string> tokenizePath (std::string *s*) `[inline]`

Tokenize a path (separator is .)

Parameters

s	String to tokenize

Returns

vector of tokens

Definition at line 26 of file XMLConfDocument.h.

9.10 XMLConf/XMLConfParser.cpp File Reference

```
#include "XMLConfParser.h"
#include <iostream>
#include <string>
#include <cstring>
```

9.11 XMLConf/XMLConfParser.h File Reference

```
#include <exception>
#include <stdexcept>
#include <sstream>
#include <set>
#include "XMLConfDocument.h"
```

Classes

- class XMLConfParser
- class XMLConfParserFatalException

9.12 XMLConf/XMLConfVersion.h File Reference

Macros

- #define XMLCONF_VERSION 1.0.0

9.12.1 Macro Definition Documentation

9.12.1.1 #define XMLCONF_VERSION 1.0.0

Definition at line 11 of file XMLConfVersion.h.

9.13 XMLConf/XMLConfWriter.cpp File Reference

```
#include "XMLConfWriter.h"
#include <sstream>
#include <iostream>
```

9.14 XMLConf/XMLConfWriter.h File Reference

```
#include <libxml/xmlmemory.h>
#include <libxml/xmlwriter.h>
#include <string>
#include <vector>
#include "XMLConfDocument.h"
```

Classes

- class XMLConfWriter

9.15 XMLConf/xmlstring.h File Reference

Macros

- #define XMLSTRING 100

Typedefs

- typedef char xmlchar

9.15.1 Macro Definition Documentation

9.15.1.1 #define XMLSTRING 100

Definition at line 6 of file xmlstring.h.

9.15.2 Typedef Documentation

9.15.2.1 typedef char **xmlchar**

Definition at line 7 of file xmlstring.h.

9.16 xmlpp.py File Reference

Namespaces

- xmlpp

Functions

- def xmlpp.parseArgs
- def xmlpp.prepareProxy
- def xmlpp.preProcessFile
- def xmlpp.printHelp
- def xmlpp.test

Variables

- tuple xmlpp.__descr__ = ("""Script for XML proxy""")
- string xmlpp.__version__ = "1.0.0"
- list xmlpp.filePath = sys.argv[1]
- string xmlpp.topStruct = ""

Index