doc

Generated by Doxygen 1.8.12

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 Converter Class Reference

Public Member Functions

- Converter (std::string _dir_path, std::string _ret_path)
- void Convert (std::string choice)

  Converts XY ANSYS FLUENT files to XML or JSON NanoDome format.

3.1.1 Detailed Description

Definition at line 21 of file converter.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Converter()

```
Converter::Converter (
            std::string _dir_path,
            std::string _ret_path )
```

Contructor

Parameters

std::string	_dir_path: directory path

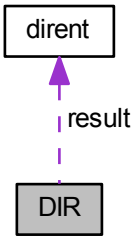
Definition at line 3 of file converter.cpp.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/converter.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/converter.cpp

3.2 DIR Struct Reference

Collaboration diagram for DIR:



Public Attributes

- handle_type **handle**
- struct _finddata_t **info**
- struct dirent **result**
- char * **name**

3.2.1 Detailed Description

Definition at line 24 of file dirent.cpp.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/win_dir_libs/dirent.cpp

3.3 dirent Struct Reference

Public Attributes

- char * **d_name**

3.3.1 Detailed Description

Definition at line 21 of file dirent.h.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/win_dir_libs/dirent.h

3.4 tinyxml2::DynArray< T, INITIAL_SIZE > Class Template Reference

Public Member Functions

- void **Clear** ()
- void **Push** (T t)
- T * **PushArr** (int count)
- T **Pop** ()
- void **PopArr** (int count)
- bool **Empty** () const
- T & **operator[ ]** (int i)
- const T & **operator[ ]** (int i) const
- const T & **PeekTop** () const
- int **Size** () const
- int **Capacity** () const
- const T * **Mem** () const
- T * **Mem** ()

3.4.1 Detailed Description

template<class T, int INITIAL_SIZE>
class tinyxml2::DynArray< T, INITIAL_SIZE >

Definition at line 184 of file tinyxml2.h.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h

3.5 tinyxml2::Entity Struct Reference

Public Attributes

- const char * **pattern**
- int **length**
- char **value**

3.5.1 Detailed Description

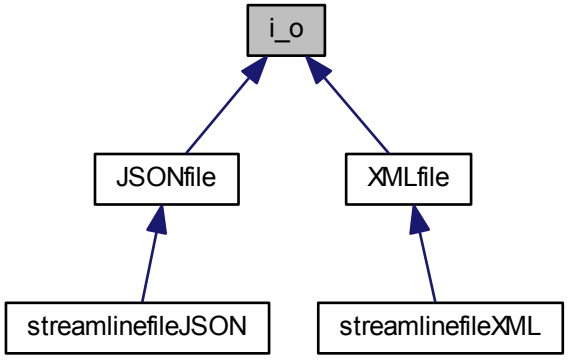
Definition at line 122 of file tinyxml2.cpp.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

3.6 i_o Class Reference

Inheritance diagram for i_o:



Public Member Functions

- i_o ()
  Contructor.
- void log_entry (std::string _entry)
- void error_entry_not_blocking (std::string _error)
- void error_entry_blocking (std::string _error)

3.6.1 Detailed Description

Definition at line 8 of file i_o.h.

3.6.2 Member Function Documentation

3.6.2.1 error_entry_blocking()

```
void i_o::error_entry_blocking (
            std::string _error )
```

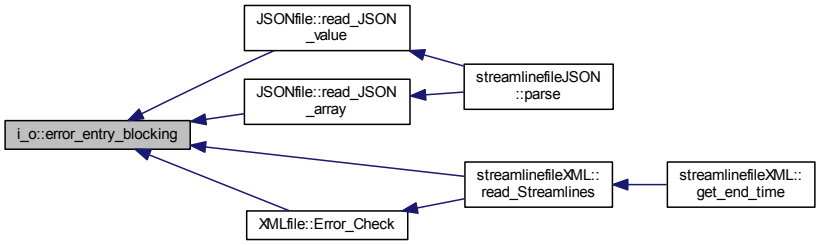
function for writing to the error output (blocking)

Parameters

std::string	error: error description

Definition at line 21 of file i_o.cpp.

Here is the caller graph for this function:



3.6.2.2 error_entry_not_blocking()

```
void i_o::error_entry_not_blocking (
            std::string _error )
```

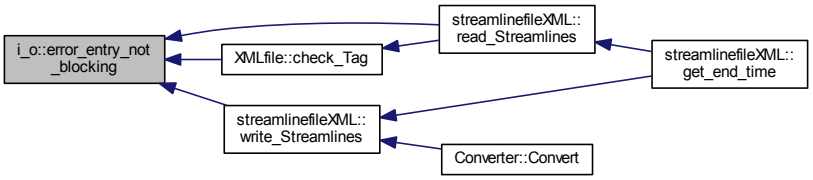
function for writing to the error output (blocking)

Parameters

std::string	error: error description

Definition at line 13 of file i_o.cpp.

Here is the caller graph for this function:



3.6.2.3 log_entry()

```
void i_o::log_entry (
            std::string _entry )
```

function for writing to the log file

Parameters

std::string	entry: entry to write

Definition at line 6 of file i_o.cpp.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/i_o.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/i_o.cpp

3.7 JSON Class Reference

Static Public Member Functions

- static JSONValue * Parse (const char *data)
- static JSONValue * Parse (const wchar_t *data)
- static std::wstring Stringify (const JSONValue *value)

Static Protected Member Functions

- static bool SkipWhitespace (const wchar_t **data)
- static bool ExtractString (const wchar_t **data, std::wstring &str)
- static double ParseInt (const wchar_t **data)
- static double ParseDecimal (const wchar_t **data)

Friends

- class **JSONValue**

3.7.1 Detailed Description

Definition at line 95 of file JSON.h.

3.7.2 Member Function Documentation

3.7.2.1 ExtractString()

```
bool JSON::ExtractString (
            const wchar_t ** data,
            std::wstring & str )  [static], [protected]
```

Extracts a JSON String as defined by the spec - "<some chars>" Any escaped characters are swapped out for their unescaped values

protected

Parameters

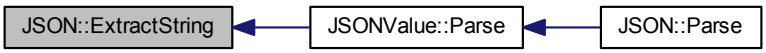
*wchar_t**	data Pointer to a wchar_t* that contains the JSON text
std::wstring&	str Reference to a std::wstring to receive the extracted string

Returns

bool Returns true on success, false on failure

Definition at line 152 of file JSON.cpp.

Here is the caller graph for this function:



3.7.2.2 Parse() [1/2]

```
JSONValue * JSON::Parse (
            const char * data )  [static]
```

Parses a complete JSON encoded string This is just a wrapper around the UNICODE Parse().

public

Parameters

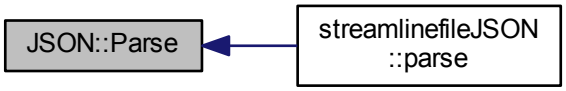
char∗	data The JSON text

Returns

JSONValue∗ Returns a JSON Value representing the root, or NULL on error

Definition at line 47 of file JSON.cpp.

Here is the caller graph for this function:



3.7.2.3 Parse() [2/2]

```
JSONValue * JSON::Parse (
            const wchar_t * data )  [static]
```

Parses a complete JSON encoded string (UNICODE input version)

public

Parameters

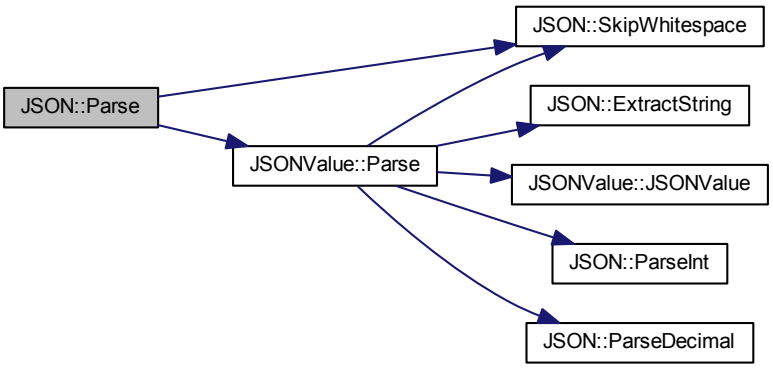
*wchar_t**	data The JSON text

Returns

JSONValue* Returns a JSON Value representing the root, or NULL on error

Definition at line 85 of file JSON.cpp.

Here is the call graph for this function:



3.7.2.4 ParseDecimal()

```
double JSON::ParseDecimal (
            const wchar_t ** data )  [static], [protected]
```

Parses some text as though it is a decimal

protected

Parameters

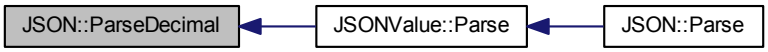
*wchar_t**	data Pointer to a wchar_t* that contains the JSON text

Returns

double Returns the double value of the decimal found

Definition at line 269 of file JSON.cpp.

Here is the caller graph for this function:



3.7.2.5 ParseInt()

```
double JSON::ParseInt (
            const wchar_t ** data )  [static], [protected]
```

Parses some text as though it is an integer

protected

Parameters

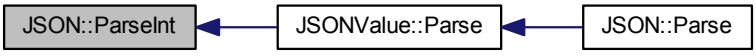
*wchar_t***	data Pointer to a wchar_t* that contains the JSON text

Returns

double Returns the double value of the number found

Definition at line 251 of file JSON.cpp.

Here is the caller graph for this function:



3.7.2.6 SkipWhitespace()

```
bool JSON::SkipWhitespace (
            const wchar_t ** data )  [static], [protected]
```

Skips over any whitespace characters (space, tab, or
) defined by the JSON spec

protected

Parameters

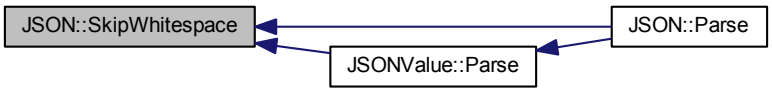
wchar_t∗∗	data Pointer to a wchar_t∗ that contains the JSON text

Returns

bool Returns true if there is more data, or false if the end of the text was reached

Definition at line 133 of file JSON.cpp.

Here is the caller graph for this function:



3.7.2.7 Stringify()

```
std::wstring JSON::Stringify (
             const JSONValue * value )  [static]
```

Turns the passed in JSONValue into a JSON encode string

public

Parameters

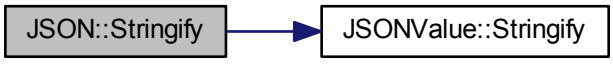
JSONValue∗	value The root value

Returns

std::wstring Returns a JSON encoded string representation of the given value

Definition at line 116 of file JSON.cpp.

Here is the call graph for this function:

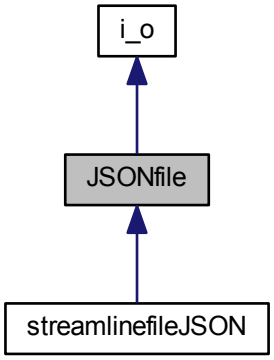


The documentation for this class was generated from the following files:

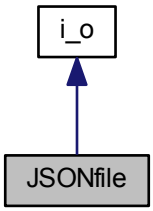
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/JSON/JSON.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/JSON/JSON.cpp

3.8 JSONfile Class Reference

Inheritance diagram for JSONfile:



Collaboration diagram for JSONfile:



Protected Member Functions

- template<class T >
  JSONArray get_JSON_array (std::vector< T > &vect, int dim)
- template<class T >
  void read_JSON_value (JSONObject j_obj, std::string tag, T &val)
- template<class T >
  void read_JSON_array (JSONObject j_obj, std::string tag, std::vector< T > &val)

Protected Attributes

- std::string JSON_file_path

  path to the JSON file
- JSONObject **root**

Additional Inherited Members

3.8.1 Detailed Description

Definition at line 12 of file JSONFile.h.

3.8.2 Member Function Documentation

3.8.2.1 get_JSON_array()

```
template<class T >
JSONArray JSONfile::get_JSON_array (
            std::vector< T > & vect,
            int dim )  [inline], [protected]
```

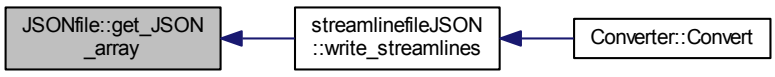
Template class for creating JSON array from std array (Implemented here because are template)

Parameters

T&	vect: vector to convert (numerical)
int	dim: vector lenght

Definition at line 27 of file JSONFile.h.

Here is the caller graph for this function:



3.8.2.2 read_JSON_array()

```
template<class T >
void JSONfile::read_JSON_array (
            JSONObject j_obj,
            std::string tag,
            std::vector< T > & val )  [inline], [protected]
```

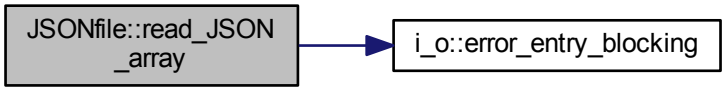
Template class for managing JSON arrays retrieving from file (Implemented here because are template)

Parameters

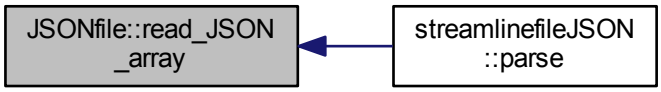
JSONObject	j_obj: inspected JSON Object
std::string	tag: tag searched
T&	val: return vector (numerical, int or float)

Definition at line 60 of file JSONFile.h.

Here is the call graph for this function:



Here is the caller graph for this function:



3.8.2.3 read_JSON_value()

```
template<class T >
void JSONfile::read_JSON_value (
             JSONObject j_obj,
             std::string tag,
             T & val )  [inline], [protected]
```

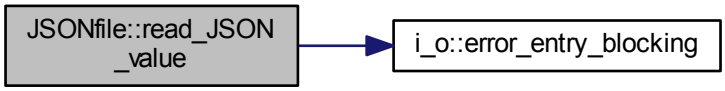
Template class for managing JSON values retrieving from file (Implemented here because are template)

Parameters

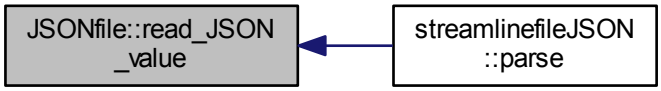
JSONObject	j_obj: inspected JSON Object
std::string	tag: tag searched
T&	val: return value (numerical, int or float)

Definition at line 42 of file JSONFile.h.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/JSONFile.h

3.9 JSONValue Class Reference

Public Member Functions

- JSONValue ()
- JSONValue (const wchar_t *m_char_value)
- JSONValue (const std::wstring &m_string_value)
- JSONValue (bool m_bool_value)
- JSONValue (double m_number_value)
- JSONValue (int m_integer_value)
- JSONValue (const JSONArray &m_array_value)
- JSONValue (const JSONObject &m_object_value)
- JSONValue (const JSONValue &m_source)
- ~JSONValue ()
- bool IsNull () const
- bool IsString () const
- bool IsBool () const
- bool IsNumber () const
- bool IsArray () const
- bool IsObject () const
- const std::wstring & AsString () const

- bool AsBool () const
- double AsNumber () const
- const JSONArray & AsArray () const
- const JSONObject & AsObject () const
- std::size_t CountChildren () const
- bool HasChild (std::size_t index) const
- JSONValue * Child (std::size_t index)
- bool HasChild (const wchar_t *name) const
- JSONValue * Child (const wchar_t *name)
- std::vector< std::wstring > ObjectKeys () const
- std::wstring Stringify (bool const prettyprint=false) const

Static Protected Member Functions

- static JSONValue * Parse (const wchar_t **data)

Friends

- class **JSON**

3.9.1 Detailed Description

Definition at line 37 of file JSONValue.h.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 JSONValue() [1/9]

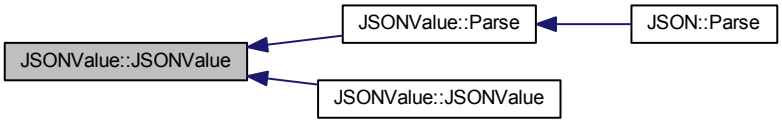
```
JSONValue::JSONValue ( )
```

Basic constructor for creating a JSON Value of type NULL

public

Definition at line 313 of file JSONValue.cpp.

Here is the caller graph for this function:



3.9.2.2 JSONValue() [2/9]

```
JSONValue::JSONValue (
            const wchar_t * m_char_value )
```

Basic constructor for creating a JSON Value of type String

public

Parameters

*wchar_t**	m_char_value The string to use as the value

Definition at line 325 of file JSONValue.cpp.

3.9.2.3 JSONValue() [3/9]

```
JSONValue::JSONValue (
            const std::wstring & m_string_value )
```

Basic constructor for creating a JSON Value of type String

public

Parameters

std::wstring	m_string_value The string to use as the value

Definition at line 338 of file JSONValue.cpp.

3.9.2.4 JSONValue() [4/9]

```
JSONValue::JSONValue (
            bool m_bool_value )
```

Basic constructor for creating a JSON Value of type Bool

public

Parameters

bool	m_bool_value The bool to use as the value

Definition at line 351 of file JSONValue.cpp.

3.9.2.5 JSONValue() [5/9]

```
JSONValue::JSONValue (
            double m_number_value )
```

Basic constructor for creating a JSON Value of type Number

public

Parameters

double	m_number_value The number to use as the value

Definition at line 364 of file JSONValue.cpp.

3.9.2.6 JSONValue() [6/9]

```
JSONValue::JSONValue (
            int m_integer_value )
```

Basic constructor for creating a JSON Value of type Number

public

Parameters

int	m_integer_value The number to use as the value

Definition at line 377 of file JSONValue.cpp.

3.9.2.7 JSONValue() [7/9]

```
JSONValue::JSONValue (
            const JSONArray & m_array_value )
```

Basic constructor for creating a JSON Value of type Array

public

Parameters

JSONArray	m_array_value The JSONArray to use as the value

Definition at line 390 of file JSONValue.cpp.

3.9.2.8 JSONValue() [8/9]

```
JSONValue::JSONValue (
            const JSONObject & m_object_value )
```

Basic constructor for creating a JSON Value of type Object

public

Parameters

JSONObject	m_object_value The JSONObject to use as the value

Definition at line 403 of file JSONValue.cpp.

3.9.2.9 JSONValue() [9/9]

```
JSONValue::JSONValue (
            const JSONValue & m_source )
```

Copy constructor to perform a deep copy of array / object values

public

Parameters

JSONValue	m_source The source JSONValue that is being copied

Definition at line 416 of file JSONValue.cpp.

Here is the call graph for this function:

3.9.2.10 ~JSONValue()

```
JSONValue::~JSONValue ( )
```

The destructor for the JSON Value object Handles deleting the objects in the array or the object value

public

Definition at line 469 of file JSONValue.cpp.

3.9.3 Member Function Documentation

3.9.3.1 AsArray()

```
const JSONArray & JSONValue::AsArray ( ) const
```

Retrieves the Array value of this JSONValue Use IsArray() before using this method.

public

Returns

JSONArray Returns the array value

Definition at line 612 of file JSONValue.cpp.

3.9.3.2 AsBool()

```
bool JSONValue::AsBool ( ) const
```

Retrieves the Bool value of this JSONValue Use IsBool() before using this method.

public

Returns

bool Returns the bool value

Definition at line 586 of file JSONValue.cpp.

3.9.3.3 AsNumber()

```
double JSONValue::AsNumber ( ) const
```

Retrieves the Number value of this JSONValue Use IsNumber() before using this method.

public

Returns

double Returns the number value

Definition at line 599 of file JSONValue.cpp.

3.9.3.4 AsObject()

```
const JSONObject & JSONValue::AsObject ( ) const
```

Retrieves the Object value of this JSONValue Use IsObject() before using this method.

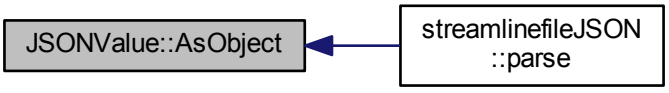
public

Returns

JSONObject Returns the object value

Definition at line 625 of file JSONValue.cpp.

Here is the caller graph for this function:



3.9.3.5 AsString()

```
const std::wstring & JSONValue::AsString ( ) const
```

Retrieves the String value of this JSONValue Use IsString() before using this method.

public

Returns

std::wstring Returns the string value

Definition at line 573 of file JSONValue.cpp.

3.9.3.6 Child() [1/2]

```
JSONValue * JSONValue::Child (
            std::size_t index )
```

Retrieves the child of this JSONValue at the given index. Use IsArray() before using this method.

public

Returns

JSONValue* Returns JSONValue at the given index or NULL if it doesn't exist.

Definition at line 681 of file JSONValue.cpp.

3.9.3.7 Child() [2/2]

```
JSONValue * JSONValue::Child (
            const wchar_t * name )
```

Retrieves the child of this JSONValue at the given key. Use IsObject() before using this method.

public

Returns

JSONValue* Returns JSONValue for the given key in the object or NULL if it doesn't exist.

Definition at line 722 of file JSONValue.cpp.

3.9.3.8 CountChildren()

```
std::size_t JSONValue::CountChildren ( ) const
```

Retrieves the number of children of this JSONValue. This number will be 0 or the actual number of children if IsArray() or IsObject().

public

Returns

The number of children.

Definition at line 639 of file JSONValue.cpp.

3.9.3.9 HasChild() [1/2]

```
bool JSONValue::HasChild (
            std::size_t index ) const
```

Checks if this JSONValue has a child at the given index. Use IsArray() before using this method.

public

Returns

bool Returns true if the array has a value at the given index.

Definition at line 660 of file JSONValue.cpp.

3.9.3.10 HasChild() [2/2]

```
bool JSONValue::HasChild (
            const wchar_t * name ) const
```

Checks if this JSONValue has a child at the given key. Use IsObject() before using this method.

public

Returns

bool Returns true if the object has a value at the given key.

Definition at line 701 of file JSONValue.cpp.

3.9.3.11 IsArray()

```
bool JSONValue::IsArray ( ) const
```

Checks if the value is an Array

public

Returns

bool Returns true if it is an Array value, false otherwise

Definition at line 548 of file JSONValue.cpp.

3.9.3.12 IsBool()

```
bool JSONValue::IsBool ( ) const
```

Checks if the value is a Bool

public

Returns

bool Returns true if it is a Bool value, false otherwise

Definition at line 524 of file JSONValue.cpp.

3.9.3.13 IsNull()

```
bool JSONValue::IsNull ( ) const
```

Checks if the value is a NULL

public

Returns

bool Returns true if it is a NULL value, false otherwise

Definition at line 500 of file JSONValue.cpp.

3.9.3.14 IsNumber()

```
bool JSONValue::IsNumber ( ) const
```

Checks if the value is a Number

public

Returns

bool Returns true if it is a Number value, false otherwise

Definition at line 536 of file JSONValue.cpp.

3.9.3.15 IsObject()

```
bool JSONValue::IsObject ( ) const
```

Checks if the value is an Object

public

Returns

bool Returns true if it is an Object value, false otherwise

Definition at line 560 of file JSONValue.cpp.

3.9.3.16 IsString()

```
bool JSONValue::IsString ( ) const
```

Checks if the value is a String

public

Returns

bool Returns true if it is a String value, false otherwise

Definition at line 512 of file JSONValue.cpp.

3.9.3.17 ObjectKeys()

```
std::vector< std::wstring > JSONValue::ObjectKeys ( ) const
```

Retrieves the keys of the JSON Object or an empty vector if this value is not an object.

public

Returns

std::vector<std::wstring> A vector containing the keys.

Definition at line 743 of file JSONValue.cpp.

3.9.3.18 Parse()

```
JSONValue * JSONValue::Parse (
            const wchar_t ** data )  [static], [protected]
```

Parses a JSON encoded value to a JSONValue object

protected

Parameters

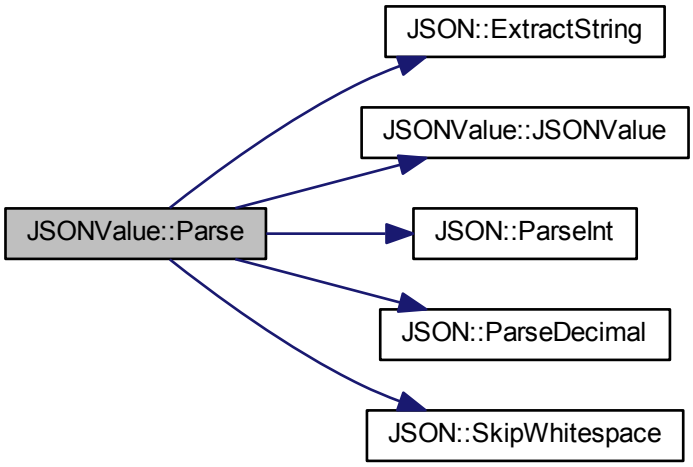
*wchar_t***	data Pointer to a wchar_t* that contains the data

Returns

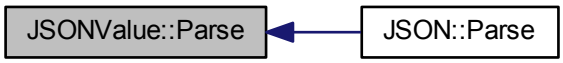
JSONValue* Returns a pointer to a JSONValue object on success, NULL on error

Definition at line 53 of file JSONValue.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.9.3.19 Stringify()

```
std::wstring JSONValue::Stringify (
            bool const prettyprint = false ) const
```

Creates a JSON encoded string for the value with all necessary characters escaped

public

Parameters

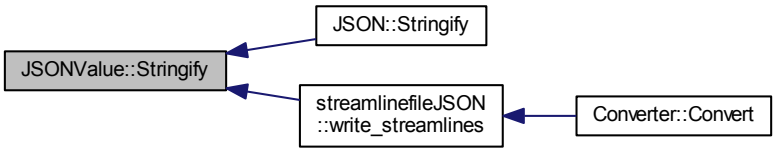
bool	prettyprint Enable prettyprint

Returns

std::wstring Returns the JSON string

Definition at line 770 of file JSONValue.cpp.

Here is the caller graph for this function:



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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/JSON/JSONValue.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/JSON/JSONValue.cpp

3.10 tinyxml2::LongFitsIntoSizeTMinusOne< bool > Struct Template Reference

Public Member Functions

- template<>
  bool **Fits** (unsigned long)

Static Public Member Functions

- static bool **Fits** (unsigned long value)

3.10.1 Detailed Description

template<bool = (sizeof(unsigned long) >= sizeof(size_t))>
struct tinyxml2::LongFitsIntoSizeTMinusOne< bool >

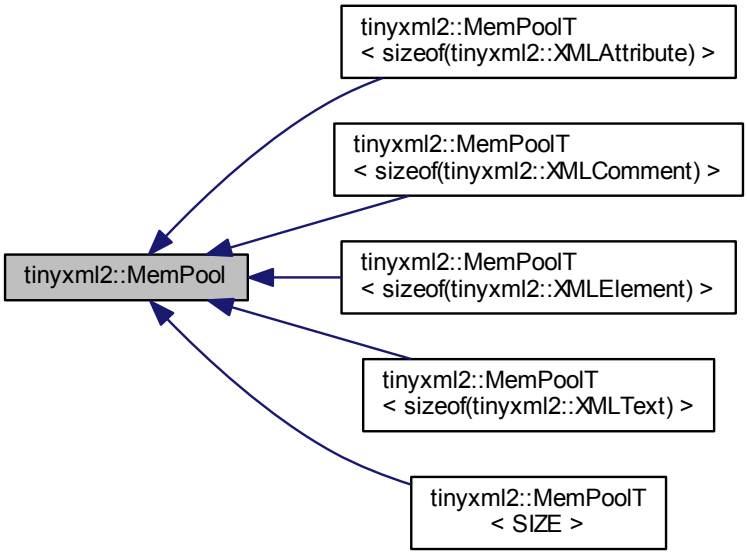
Definition at line 1927 of file tinyxml2.cpp.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

3.11 tinyxml2::MemPool Class Reference

Inheritance diagram for tinyxml2::MemPool:



Public Member Functions

- virtual int **ItemSize** () const =0
- virtual void * **Alloc** ()=0
- virtual void **Free** (void *)=0
- virtual void **SetTracked** ()=0
- virtual void **Clear** ()=0

3.11.1 Detailed Description

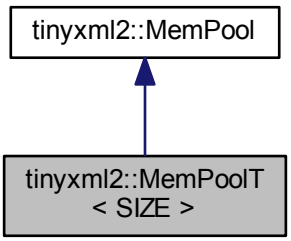
Definition at line 297 of file tinyxml2.h.

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

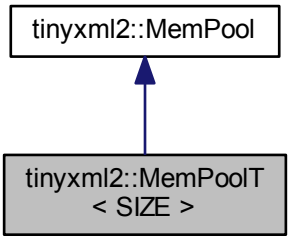
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h

3.12 tinyxml2::MemPoolT< SIZE > Class Template Reference

Inheritance diagram for tinyxml2::MemPoolT< SIZE >:



Collaboration diagram for tinyxml2::MemPoolT< SIZE >:



Public Types

- enum { **COUNT** = (4∗1024)/SIZE }

Public Member Functions

- void **Clear** ()
- virtual int **ItemSize** () const
- int **CurrentAllocs** () const
- virtual void ∗ **Alloc** ()
- virtual void **Free** (void ∗mem)
- void **Trace** (const char ∗name)
- void **SetTracked** ()
- int **Untracked** () const

3.12.1 Detailed Description

template<int SIZE>
class tinyxml2::MemPoolT< SIZE >

Definition at line 315 of file tinyxml2.h.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h

3.13 Raw_data Class Reference

Public Member Functions

- Raw_data ()
  Blank Contructor.
- int get_n_streams () const
  Returns the number of streamlines tracked.
- void init_times (std::vector< std::vector< double > > &time_v)
- void init_temperatures (std::vector< std::vector< double > > &temp_v)
- void init_pressures (std::vector< std::vector< double > > &press_v)
- void init_x (std::vector< std::vector< double > > &x_v)
- void init_y (std::vector< std::vector< double > > &y_v)
- void init_z (std::vector< std::vector< double > > &z_v)
- void init_molar_c (std::vector< std::vector< double > > &molar_v)
- std::vector< double > **get_times** (int idx) const
- std::vector< double > **get_temperatures** (int idx) const
- std::vector< double > **get_molar_cs** (int idx) const
- std::vector< double > **get_x** (int idx) const
- std::vector< double > **get_y** (int idx) const
- std::vector< double > **get_z** (int idx) const
- std::vector< double > **get_pressures** (int idx) const
- std::list< Species > **get_species** () const

Public Attributes

- std::vector< std::vector< double > > times
  raw data from the times extracted from XY file from ANSYS FLUENT
- std::vector< std::vector< double > > temperatures
  raw data from the tenperatures extracted from XY file from ANSYS FLUENT
- std::vector< std::vector< double > > pressures
  raw data from the pressures extracted from XY file from ANSYS FLUENT
- std::vector< std::vector< double > > molar_cs
  raw data from the species molar concentrations extracted from XY file from ANSYS FLUENT
- std::vector< std::vector< double > > x
  raw data from the x coordinates extracted from XY file from ANSYS FLUENT
- std::vector< std::vector< double > > y
  raw data from the y coordinates extracted from XY file from ANSYS FLUENT
- std::vector< std::vector< double > > z
  raw data from the z coordinates extracted from XY file from ANSYS FLUENT
- std::list< Species > species
  raw data from the species extracted from XY file from ANSYS FLUENT

3.13.1 Detailed Description

Definition at line 10 of file raw_data.h.

3.13.2 Member Function Documentation

3.13.2.1 init_molar_c()

```
void Raw_data::init_molar_c (
            std::vector< std::vector< double > > & molar_v )  [inline]
```

Initiate the molar concentration samples arrays

Parameters

std::vector<	std::vector< double > > molar_v: molar concentration for each species extracted from XY files

Definition at line 85 of file raw_data.h.

3.13.2.2 init_pressures()

```
void Raw_data::init_pressures (
            std::vector< std::vector< double > > & press_v )  [inline]
```

Initiate the pressures samples arrays

Parameters

std::vector<	std::vector< double > > press_v: pressures samples extracted from XY files

Definition at line 69 of file raw_data.h.

3.13.2.3 init_temperatures()

```
void Raw_data::init_temperatures (
            std::vector< std::vector< double > > & temp_v )  [inline]
```

Initiate the temperatures samples arrays

Parameters

std::vector<	std::vector< double > > temp_v: temperatures samples extracted from XY files

Definition at line 65 of file raw_data.h.

3.13.2.4 init_times()

```
void Raw_data::init_times (
            std::vector< std::vector< double > > & time_v )  [inline]
```

Initiate the time samples array

Parameters

std::vector<	std::vector< double > > time_v: time samples extracted from XY files

Definition at line 61 of file raw_data.h.

3.13.2.5 init_x()

```
void Raw_data::init_x (
            std::vector< std::vector< double > > & x_v )  [inline]
```

Initiate the x coordinates arrays

Parameters

std::vector<	std::vector< double > > x_v: x coordinates extracted from XY files

Definition at line 73 of file raw_data.h.

3.13.2.6 init_y()

```
void Raw_data::init_y (
            std::vector< std::vector< double > > & y_v )  [inline]
```

Initiate the y coordinates arrays

Parameters

std::vector<	std::vector< double > > y_v: y coordinates extracted from XY files

Definition at line 77 of file raw_data.h.

3.13.2.7 init_z()

```
void Raw_data::init_z (
            std::vector< std::vector< double > > & z_v )  [inline]
```

Initiate the z coordinates arrays

Parameters

std::vector<	std::vector< double > > z_v: z coordinates extracted from XY files

Definition at line 81 of file raw_data.h.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/raw_data.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/raw_data.cpp

3.14 Species Class Reference

Public Member Functions

- Species (std::string _formula)
  get formula
- std::string get_formula () const
  species mass [kg]
- double get_mass () const
  species L-J sigma [m]
- double get_sigma () const
  species L-J epsilon [J]
- double get_epsilon () const
- double s_ten (double T) const
- double p_sat (double T) const
- double n_sat (double T) const
  molecular volume based on liquid density [m3]
- double m_volume () const
  molecular surface based on liquid density and spherical assumption [m2]
- double **m_surface** () const
- double **get_bulk_density** (double T) const

3.14.1 Detailed Description

Definition at line 10 of file species.h.

3.14.2 Member Function Documentation

3.14.2.1 get_epsilon()

```
double Species::get_epsilon ( ) const  [inline]
```

bulk material surface tension [N/m]

Parameters

T	temperature [K]

Definition at line 52 of file species.h.

3.14.2.2 p_sat()

```
double Species::p_sat (
            double T ) const  [inline]
```

saturation density [#/m3]

Parameters

T	temperature [K]

Definition at line 60 of file species.h.

Here is the caller graph for this function:

3.14.2.3 s_ten()

```
double Species::s_ten (
            double T ) const  [inline]
```

saturation pressure [Pa]

Parameters

T	temperature [K]

Definition at line 56 of file species.h.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/species.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/species.cpp

3.15 streamline Class Reference

Public Member Functions

- streamline ()

  Default Constructor.
- streamline (int _ID, std::vector< double > _T, std::vector< double > _P, std::vector< double > _Time, std::list< Species > _species, std::vector< double > _Molar_Conc, std::vector< double > _X, std::vector< double > _Y, std::vector< double > _Z)

  Parametric Constructor.
- std::vector< double > get_Temp () const

  Return Temperature [K] array.
- std::vector< double > get_Press () const

  Return Pressure [Pa] array.
- std::vector< double > & get_Time ()

  Return Times [sec].
- std::vector< double > get_Molar () const

  Return Molar Concentration for each species [sec].
- std::list< Species > get_Species () const

  Return Species.
- int get_ID () const

  Return Streamline ID.
- std::vector< double > get_x () const

  Return X Positions.
- std::vector< double > get_y () const

  Return Y Positions.
- std::vector< double > get_z () const

  Return Z Positions.
- void printstream ()

  Print Streamline.

3.15.1 Detailed Description

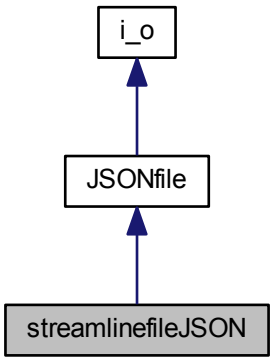
Definition at line 11 of file streamline.h.

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

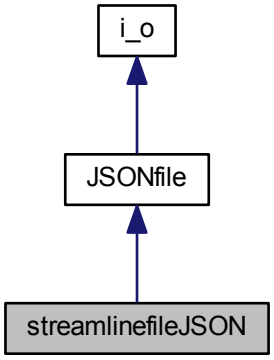
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/streamline.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/streamline.cpp

3.16 streamlinefileJSON Class Reference

Inheritance diagram for streamlinefileJSON:



Collaboration diagram for streamlinefileJSON:



Public Member Functions

- streamlinefileJSON (std::string _file)
- std::vector< streamline > parse ()

  Parse the JSON file.
- void write_streamlines (std::vector< streamline > &_streams)

  Write the JSON file starting from a set of streamlines.

Additional Inherited Members

3.16.1 Detailed Description

Definition at line 13 of file streamlinefileJSON.h.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 streamlinefileJSON()

```
streamlinefileJSON::streamlinefileJSON (
            std::string _file )
```

Contructor

Parameters

std::string	_file: path to the JSON file

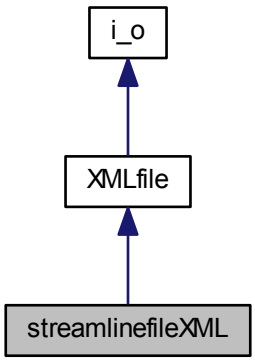
Definition at line 11 of file streamlinefileJSON.cpp.

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

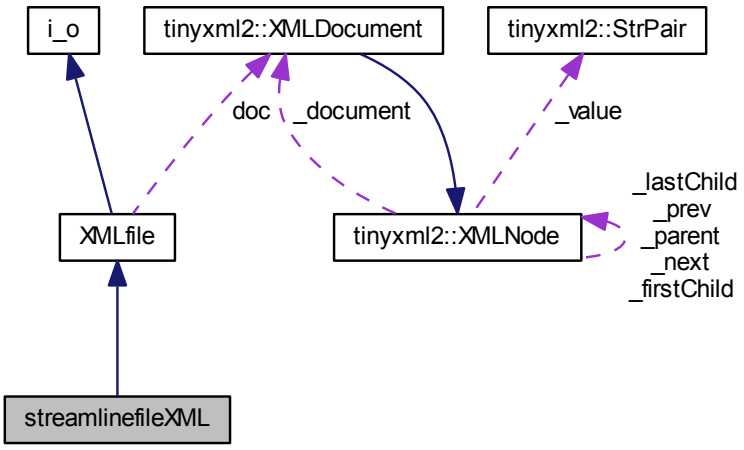
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/streamlinefileJSON.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/streamlinefileJSON.cpp

3.17 streamlinefileXML Class Reference

Inheritance diagram for streamlinefileXML:



Collaboration diagram for streamlinefileXML:



Public Member Functions

- streamlinefileXML (std::string _path, std::vector< streamline > &_streams)
- std::vector< streamline > get_streamlines () const

  Return Streamlines from file.
- double get_start_time () const

  Return start time.
- double get_end_time () const

  Return end sampling time.
- void read_Streamlines ()

  extract Streamlines from XML file
- void write_Streamlines (std::vector< streamline > &_streamlines, double _start_t, double _end_t)
- void write_Streamlines (std::string _path)
- void printstreamlines ()

  Print Streamlines.

Additional Inherited Members

3.17.1 Detailed Description

Definition at line 12 of file streamlinefileXML.h.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 streamlinefileXML()

```
streamlinefileXML::streamlinefileXML (
            std::string _path,
            std::vector< streamline > & _streams )
```

Constructor (reads and stores the XML File)

Parameters

string	path: Path of the XML file

Get XML File

Definition at line 17 of file streamlinefileXML.cpp.

3.17.3 Member Function Documentation

3.17.3.1 write_Streamlines() [1/2]

```
void streamlinefileXML::write_Streamlines (
            std::vector< streamline > & _streamlines,
            double _start_t,
            double _end_t )
```

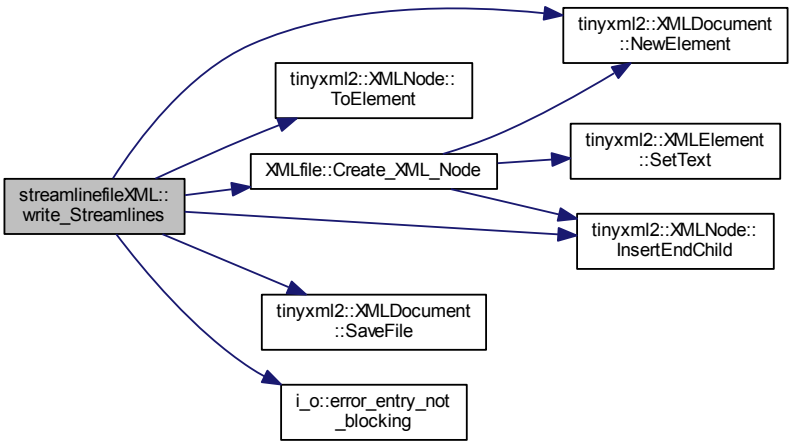
write streamlines (defined outside the object) to file

Parameters

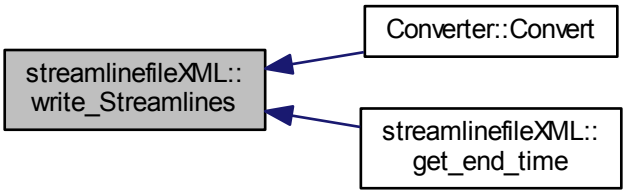
std::vector<streamline>&	streamlines reference to streamlines container
double	_start_t: sampling start time
double	_end_t: sampling end time

Definition at line 227 of file streamlinefileXML.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.17.3.2 write_Streamlines() [2/2]

```
void streamlinefileXML::write_Streamlines (
            std::string _path )
```

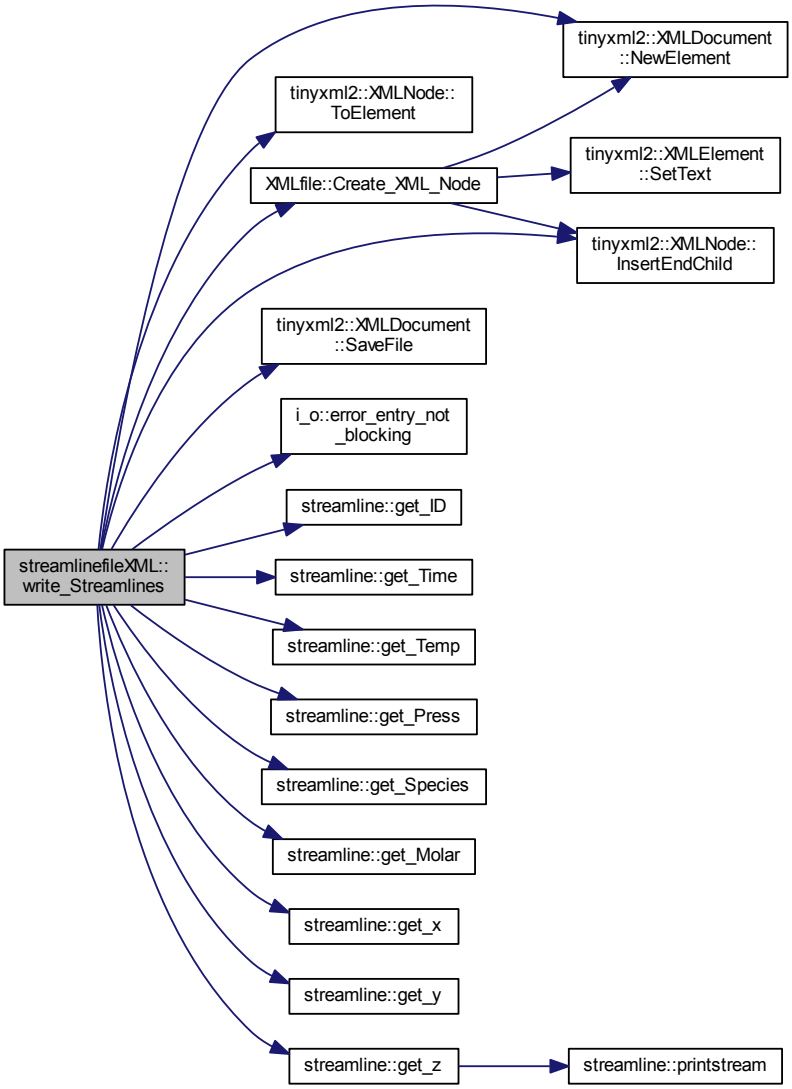
write streamlines defined inside the object to file

Parameters

std::string	_path: output file path

Definition at line 262 of file streamlinefileXML.cpp.

Here is the call graph for this function:



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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/streamlinefileXML.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/streamlinefileXML.cpp

3.18 tinyxml2::StrPair Class Reference

Public Types

- enum {
  NEEDS_ENTITY_PROCESSING = 0x01, **NEEDS_NEWLINE_NORMALIZATION** = 0x02, **NEEDS_WHIT**↩
  ESPACE_COLLAPSING = 0x04, **TEXT_ELEMENT** = NEEDS_ENTITY_PROCESSING | NEEDS_NEWL↩
  INE_NORMALIZATION,

TEXT_ELEMENT_LEAVE_ENTITIES = NEEDS_NEWLINE_NORMALIZATION, **ATTRIBUTE_NAME** = 0, **ATTRIBUTE_VALUE** = NEEDS_ENTITY_PROCESSING | NEEDS_NEWLINE_NORMALIZATION, **ATTRIBUTE_VALUE_LEAVE_ENTITIES** = NEEDS_NEWLINE_NORMALIZATION, **COMMENT** = NEEDS_NEWLINE_NORMALIZATION }

Public Member Functions

- void **Set** (char *start, char *end, int flags)
- const char * **GetStr** ()
- bool **Empty** () const
- void **SetInternedStr** (const char *str)
- void **SetStr** (const char *str, int flags=0)
- char * **ParseText** (char *in, const char *endTag, int strFlags)
- char * **ParseName** (char *in)
- void **TransferTo** (StrPair *other)

3.18.1 Detailed Description

Definition at line 116 of file tinyxml2.h.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

3.19 tinyxml2::XMLAttribute Class Reference

`#include <tinyxml2.h>`

Public Member Functions

- const char * Name () const

  The name of the attribute.
- const char * Value () const

  The value of the attribute.
- const XMLAttribute * Next () const

  The next attribute in the list.
- int IntValue () const
- unsigned UnsignedValue () const

  Query as an unsigned integer. See IntValue()
- bool BoolValue () const

  Query as a boolean. See IntValue()
- double DoubleValue () const

  Query as a double. See IntValue()
- float FloatValue () const

  Query as a float. See IntValue()
- XMLError QueryIntValue (int *value) const
- XMLError QueryUnsignedValue (unsigned int *value) const

See QueryIntValue.

- XMLError QueryBoolValue (bool ∗value) const

  See QueryIntValue.
- XMLError QueryDoubleValue (double ∗value) const

  See QueryIntValue.
- XMLError QueryFloatValue (float ∗value) const

  See QueryIntValue.
- void SetAttribute (const char ∗value)

  Set the attribute to a string value.
- void SetAttribute (int value)

  Set the attribute to value.
- void SetAttribute (unsigned value)

  Set the attribute to value.
- void SetAttribute (bool value)

  Set the attribute to value.
- void SetAttribute (double value)

  Set the attribute to value.
- void SetAttribute (float value)

  Set the attribute to value.

Friends

- class **XMLElement**

3.19.1 Detailed Description

An attribute is a name-value pair. Elements have an arbitrary number of attributes, each with a unique name.

Note

The attributes are not XMLNodes. You may only query the Next() attribute in a list.

Definition at line 1043 of file tinyxml2.h.

3.19.2 Member Function Documentation

3.19.2.1 IntValue()

```
int tinyxml2::XMLAttribute::IntValue ( ) const  [inline]
```

IntValue interprets the attribute as an integer, and returns the value. If the value isn't an integer, 0 will be returned. There is no error checking; use QueryIntValue() if you need error checking.

Definition at line 1062 of file tinyxml2.h.

3.19.2.2 QueryIntValue()

```
XMLError tinyxml2::XMLAttribute::QueryIntValue (
            int * value ) const
```

QueryIntValue interprets the attribute as an integer, and returns the value in the provided parameter. The function will return XML_NO_ERROR on success, and XML_WRONG_ATTRIBUTE_TYPE if the conversion is not successful.

Definition at line 1286 of file tinyxml2.cpp.

Here is the caller graph for this function:

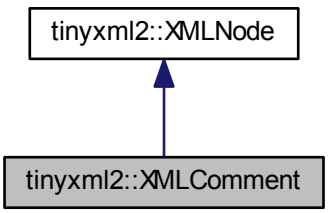
The documentation for this class was generated from the following files:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

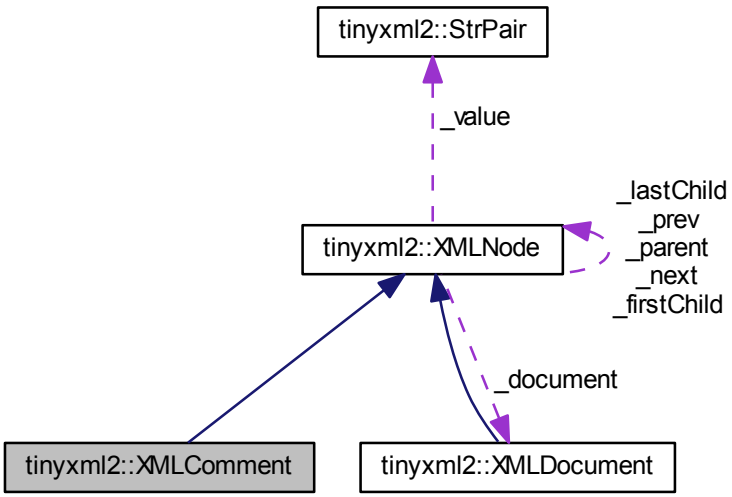
3.20 tinyxml2::XMLComment Class Reference

```
#include <tinyxml2.h>
```

Inheritance diagram for tinyxml2::XMLComment:



Collaboration diagram for tinyxml2::XMLComment:



Public Member Functions

- virtual XMLComment * ToComment ()

  Safely cast to a Comment, or null.
- virtual const XMLComment * **ToComment** () const
- virtual bool Accept (XMLVisitor *visitor) const
- virtual XMLNode * ShallowClone (XMLDocument *document) const
- virtual bool ShallowEqual (const XMLNode *compare) const

Protected Member Functions

- **XMLComment** (XMLDocument *doc)
- char * **ParseDeep** (char *, StrPair *endTag)

Friends

- class **XMLDocument**

Additional Inherited Members

3.20.1 Detailed Description

An XML Comment.

Definition at line 934 of file tinyxml2.h.

3.20.2 Member Function Documentation

3.20.2.1 Accept()

```
bool tinyxml2::XMLComment::Accept (
            XMLVisitor * visitor ) const  [virtual]
```

Accept a hierarchical visit of the nodes in the TinyXML-2 DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the XMLVisitor interface.

This is essentially a SAX interface for TinyXML-2. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML-2 is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- http://www.saxproject.org/
- http://c2.com/cgi/wiki?HierarchicalVisitorPattern

Which are both good references for "visiting".

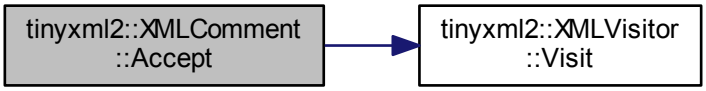
An example of using Accept():

```
XMLPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

Implements tinyxml2::XMLNode.

Definition at line 1134 of file tinyxml2.cpp.

Here is the call graph for this function:



3.20.2.2 ShallowClone()

```
XMLNode * tinyxml2::XMLComment::ShallowClone (
            XMLDocument * document ) const  [virtual]
```

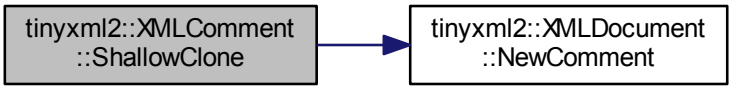
Make a copy of this node, but not its children. You may pass in a Document pointer that will be the owner of the new Node. If the 'document' is null, then the node returned will be allocated from the current Document. (this->GetDocument())

Note: if called on a XMLDocument, this will return null.

Implements tinyxml2::XMLNode.

Definition at line 1116 of file tinyxml2.cpp.

Here is the call graph for this function:



3.20.2.3 ShallowEqual()

```
bool tinyxml2::XMLComment::ShallowEqual (
            const XMLNode * compare ) const  [virtual]
```

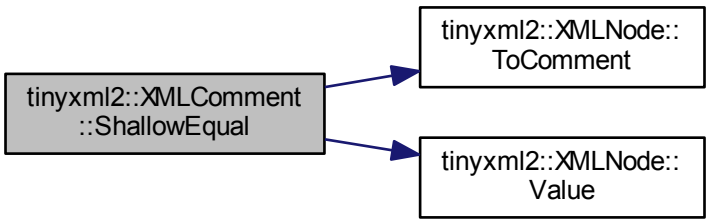
Test if 2 nodes are the same, but don't test children. The 2 nodes do not need to be in the same Document.

Note: if called on a XMLDocument, this will return false.

Implements tinyxml2::XMLNode.

Definition at line 1126 of file tinyxml2.cpp.

Here is the call graph for this function:



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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

3.21 tinyxml2::XMLConstHandle Class Reference

`#include <tinyxml2.h>`

Public Member Functions

- **XMLConstHandle** (const XMLNode *node)
- **XMLConstHandle** (const XMLNode &node)
- **XMLConstHandle** (const XMLConstHandle &ref)
- XMLConstHandle & **operator=** (const XMLConstHandle &ref)
- const XMLConstHandle **FirstChild** () const
- const XMLConstHandle **FirstChildElement** (const char *name=0) const
- const XMLConstHandle **LastChild** () const
- const XMLConstHandle **LastChildElement** (const char *name=0) const
- const XMLConstHandle **PreviousSibling** () const
- const XMLConstHandle **PreviousSiblingElement** (const char *name=0) const
- const XMLConstHandle **NextSibling** () const
- const XMLConstHandle **NextSiblingElement** (const char *name=0) const
- const XMLNode * **ToNode** () const
- const XMLElement * **ToElement** () const
- const XMLText * **ToText** () const
- const XMLUnknown * **ToUnknown** () const
- const XMLDeclaration * **ToDeclaration** () const

3.21.1 Detailed Description

A variant of the XMLHandle class for working with const XMLNodes and Documents. It is the same in all regards, except for the 'const' qualifiers. See XMLHandle for API.

Definition at line 1870 of file tinyxml2.h.

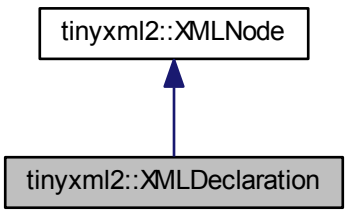
The documentation for this class was generated from the following file:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h

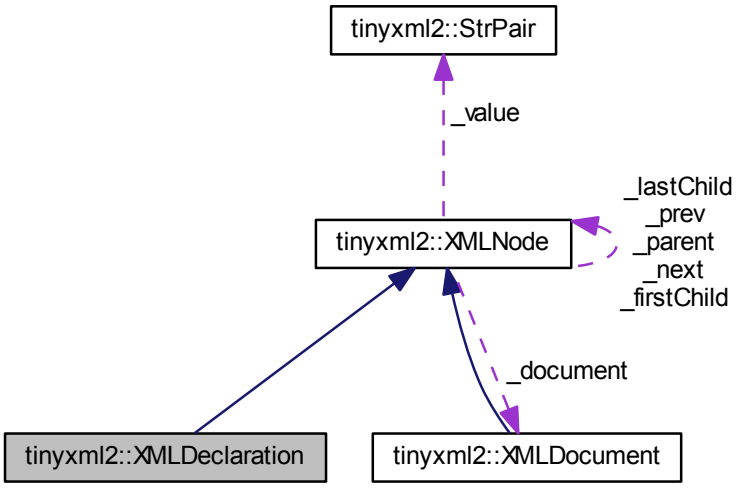
3.22 tinyxml2::XMLDeclaration Class Reference

`#include <tinyxml2.h>`

Inheritance diagram for tinyxml2::XMLDeclaration:



Collaboration diagram for tinyxml2::XMLDeclaration:



Public Member Functions

- virtual XMLDeclaration ∗ ToDeclaration ()

  Safely cast to a Declaration, or null.
- virtual const XMLDeclaration ∗ **ToDeclaration** () const
- virtual bool Accept (XMLVisitor ∗visitor) const
- virtual XMLNode ∗ ShallowClone (XMLDocument ∗document) const
- virtual bool ShallowEqual (const XMLNode ∗compare) const

Protected Member Functions

- **XMLDeclaration** (XMLDocument ∗doc)
- char ∗ **ParseDeep** (char ∗, StrPair ∗endTag)

Friends

- class **XMLDocument**

Additional Inherited Members

3.22.1 Detailed Description

In correct XML the declaration is the first entry in the file.

```
<?xml version="1.0" standalone="yes"?>
```

TinyXML-2 will happily read or write files without a declaration, however.

The text of the declaration isn't interpreted. It is parsed and written as a string.

Definition at line 973 of file tinyxml2.h.

3.22.2 Member Function Documentation

3.22.2.1 Accept()

```
bool tinyxml2::XMLDeclaration::Accept (
            XMLVisitor * visitor ) const  [virtual]
```

Accept a hierarchical visit of the nodes in the TinyXML-2 DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the XMLVisitor interface.

This is essentially a SAX interface for TinyXML-2. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML-2 is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- http://www.saxproject.org/
- http://c2.com/cgi/wiki?HierarchicalVisitorPattern

Which are both good references for "visiting".

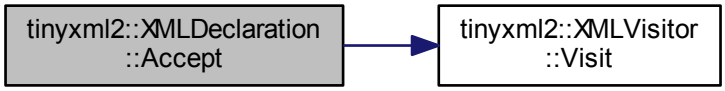
An example of using Accept():

```
XMLPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

Implements tinyxml2::XMLNode.

Definition at line 1185 of file tinyxml2.cpp.

Here is the call graph for this function:



3.22.2.2 ShallowClone()

```
XMLNode * tinyxml2::XMLDeclaration::ShallowClone (
            XMLDocument * document ) const  [virtual]
```

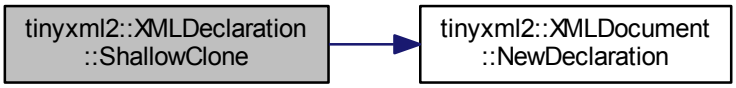
Make a copy of this node, but not its children. You may pass in a Document pointer that will be the owner of the new Node. If the 'document' is null, then the node returned will be allocated from the current Document. (this->GetDocument())

Note: if called on a XMLDocument, this will return null.

Implements tinyxml2::XMLNode.

Definition at line 1166 of file tinyxml2.cpp.

Here is the call graph for this function:



3.22.2.3 ShallowEqual()

```
bool tinyxml2::XMLDeclaration::ShallowEqual (
            const XMLNode * compare ) const  [virtual]
```

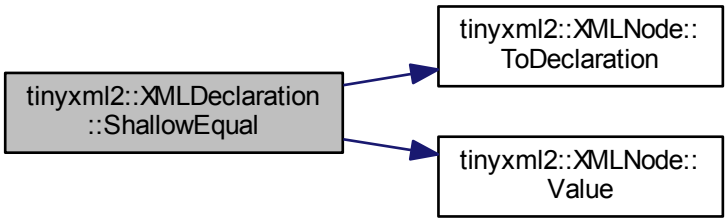
Test if 2 nodes are the same, but don't test children. The 2 nodes do not need to be in the same Document.

Note: if called on a XMLDocument, this will return false.

Implements tinyxml2::XMLNode.

Definition at line 1176 of file tinyxml2.cpp.

Here is the call graph for this function:



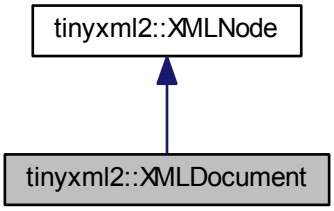
The documentation for this class was generated from the following files:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

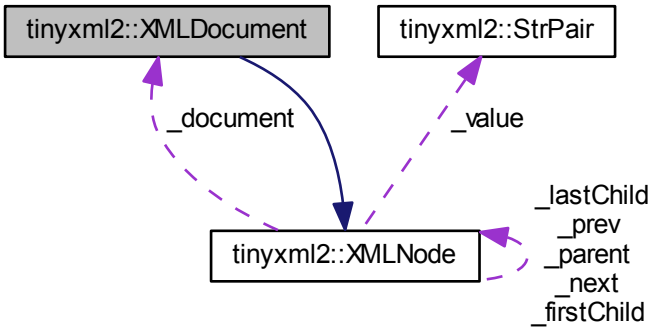
3.23 tinyxml2::XMLDocument Class Reference

`#include <tinyxml2.h>`

Inheritance diagram for tinyxml2::XMLDocument:



Collaboration diagram for tinyxml2::XMLDocument:



Public Member Functions

- XMLDocument (bool processEntities=true, Whitespace=PRESERVE_WHITESPACE)
  constructor
- virtual XMLDocument ∗ ToDocument ()
  Safely cast to a Document, or null.
- virtual const XMLDocument ∗ **ToDocument** () const
- XMLError Parse (const char ∗xml, size_t nBytes=(size_t)(-1))
- XMLError LoadFile (const char ∗filename)
- XMLError LoadFile (FILE ∗)
- XMLError SaveFile (const char ∗filename, bool compact=false)
- XMLError SaveFile (FILE ∗fp, bool compact=false)

- bool **ProcessEntities** () const
- Whitespace **WhitespaceMode** () const
- bool HasBOM () const
- void SetBOM (bool useBOM)
- XMLElement * RootElement ()
- const XMLElement * **RootElement** () const
- void Print (XMLPrinter *streamer=0) const
- virtual bool Accept (XMLVisitor *visitor) const
- XMLElement * NewElement (const char *name)
- XMLComment * NewComment (const char *comment)
- XMLText * NewText (const char *text)
- XMLDeclaration * NewDeclaration (const char *text=0)
- XMLUnknown * NewUnknown (const char *text)
- void DeleteNode (XMLNode *node)
- void **SetError** (XMLError error, const char *str1, const char *str2)
- bool Error () const

  Return true if there was an error parsing the document.
- XMLError ErrorID () const

  Return the errorID.
- const char * **ErrorName** () const
- const char * GetErrorStr1 () const

  Return a possibly helpful diagnostic location or string.
- const char * GetErrorStr2 () const

  Return a possibly helpful secondary diagnostic location or string.
- void PrintError () const

  If there is an error, print it to stdout.
- void Clear ()

  Clear the document, resetting it to the initial state.
- char * **Identify** (char *p, XMLNode **node)
- virtual XMLNode * ShallowClone (XMLDocument *) const
- virtual bool ShallowEqual (const XMLNode *) const

Friends

- class **XMLElement**

Additional Inherited Members

3.23.1 Detailed Description

A Document binds together all the functionality. It can be saved, loaded, and printed to the screen. All Nodes are connected and allocated to a Document. If the Document is deleted, all its Nodes are also deleted.

Definition at line 1518 of file tinyxml2.h.

3.23.2 Member Function Documentation

3.23.2.1 Accept()

```
bool tinyxml2::XMLDocument::Accept (
            XMLVisitor * visitor ) const  [virtual]
```

Accept a hierarchical visit of the nodes in the TinyXML-2 DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the XMLVisitor interface.

This is essentially a SAX interface for TinyXML-2. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML-2 is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- http://www.saxproject.org/
- http://c2.com/cgi/wiki?HierarchicalVisitorPattern

Which are both good references for "visiting".

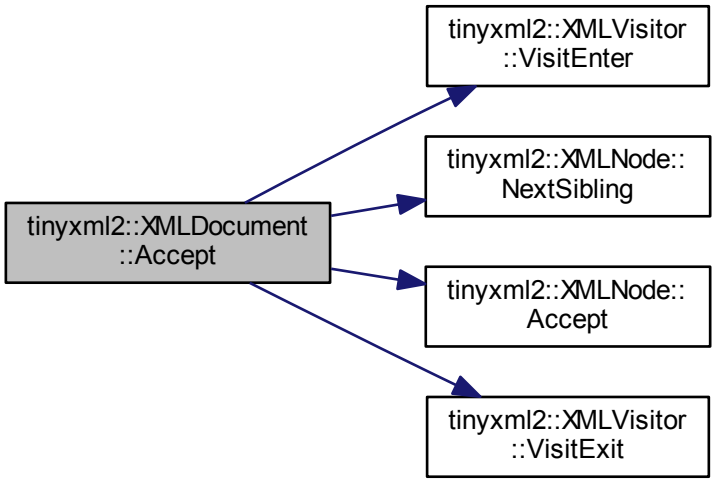
An example of using Accept():

```
XMLPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

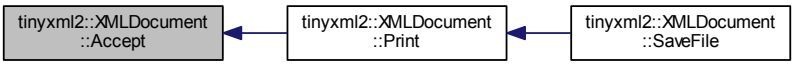
Implements tinyxml2::XMLNode.

Definition at line 685 of file tinyxml2.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



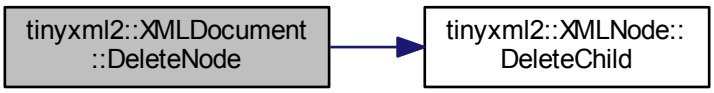
3.23.2.2 DeleteNode()

```
void tinyxml2::XMLDocument::DeleteNode (
            XMLNode * node )
```

Delete a node associated with this document. It will be unlinked from the DOM.

Definition at line 1888 of file tinyxml2.cpp.

Here is the call graph for this function:



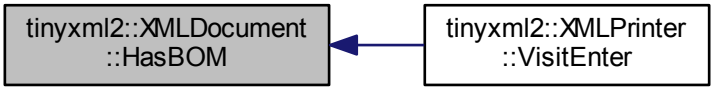
3.23.2.3 HasBOM()

```
bool tinyxml2::XMLDocument::HasBOM ( ) const  [inline]
```

Returns true if this document has a leading Byte Order Mark of UTF8.

Definition at line 1593 of file tinyxml2.h.

Here is the caller graph for this function:



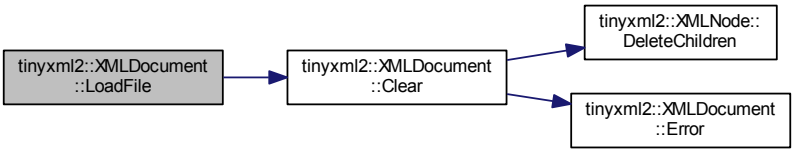
3.23.2.4 LoadFile() [1/2]

```
XMLError tinyxml2::XMLDocument::LoadFile (
            const char * filename )
```

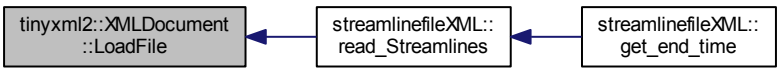
Load an XML file from disk. Returns XML_NO_ERROR (0) on success, or an errorID.

Definition at line 1906 of file tinyxml2.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.23.2.5 LoadFile() [2/2]

```
XMLError tinyxml2::XMLDocument::LoadFile (
            FILE * fp )
```

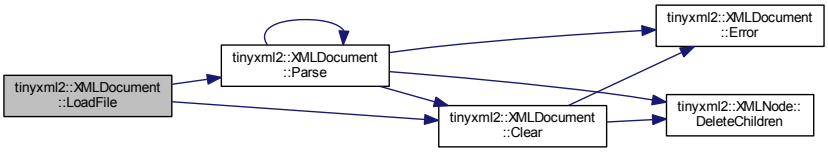
Load an XML file from disk. You are responsible for providing and closing the FILE∗.

NOTE: The file should be opened as binary ("rb") not text in order for TinyXML-2 to correctly do newline normalization.

Returns XML_NO_ERROR (0) on success, or an errorID.

Definition at line 1940 of file tinyxml2.cpp.

Here is the call graph for this function:



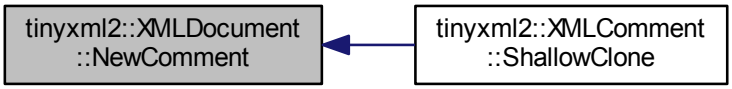
3.23.2.6 NewComment()

```
XMLComment * tinyxml2::XMLDocument::NewComment (
            const char * comment )
```

Create a new Comment associated with this Document. The memory for the Comment is managed by the Document.

Definition at line 1833 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.23.2.7 NewDeclaration()

```
XMLDeclaration * tinyxml2::XMLDocument::NewDeclaration (
            const char * text = 0 )
```

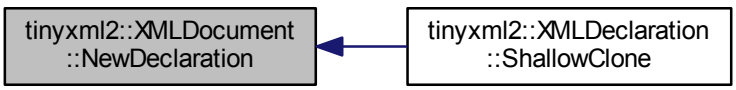
Create a new Declaration associated with this Document. The memory for the object is managed by the Document.

If the 'text' param is null, the standard declaration is used.:

```
    <?xml version="1.0" encoding="UTF-8"?>
```

Definition at line 1853 of file tinyxml2.cpp.

Here is the caller graph for this function:



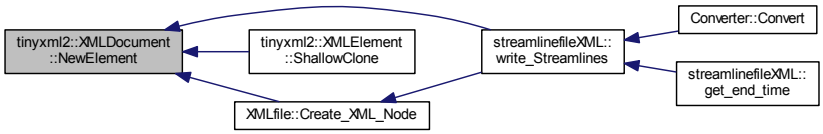
3.23.2.8 NewElement()

```
XMLElement * tinyxml2::XMLDocument::NewElement (
            const char * name )
```

Create a new Element associated with this Document. The memory for the Element is managed by the Document.

Definition at line 1823 of file tinyxml2.cpp.

Here is the caller graph for this function:



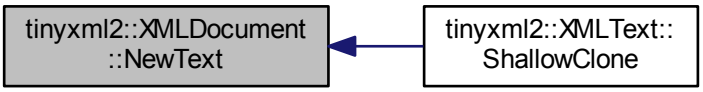
3.23.2.9 NewText()

```
XMLText * tinyxml2::XMLDocument::NewText (
            const char * text )
```

Create a new Text associated with this Document. The memory for the Text is managed by the Document.

Definition at line 1843 of file tinyxml2.cpp.

Here is the caller graph for this function:



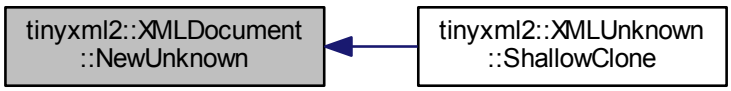
3.23.2.10 NewUnknown()

```
XMLUnknown * tinyxml2::XMLDocument::NewUnknown (
            const char * text )
```

Create a new Unknown associated with this Document. The memory for the object is managed by the Document.

Definition at line 1863 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.23.2.11 Parse()

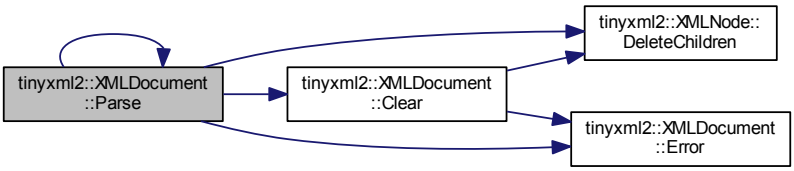
```
XMLError tinyxml2::XMLDocument::Parse (
            const char * xml,
            size_t nBytes = (size_t)(-1) )
```

Parse an XML file from a character string. Returns XML_NO_ERROR (0) on success, or an errorID.

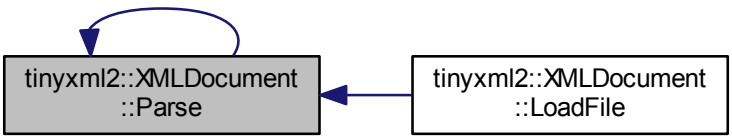
You may optionally pass in the 'nBytes', which is the number of bytes which will be parsed. If not specified, TinyX↩ ML-2 will assume 'xml' points to a null terminated string.

Definition at line 2010 of file tinyxml2.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.23.2.12 Print()

```
void tinyxml2::XMLDocument::Print (
            XMLPrinter * streamer = 0 ) const
```

Print the Document. If the Printer is not provided, it will print to stdout. If you provide Printer, this can print to a file:

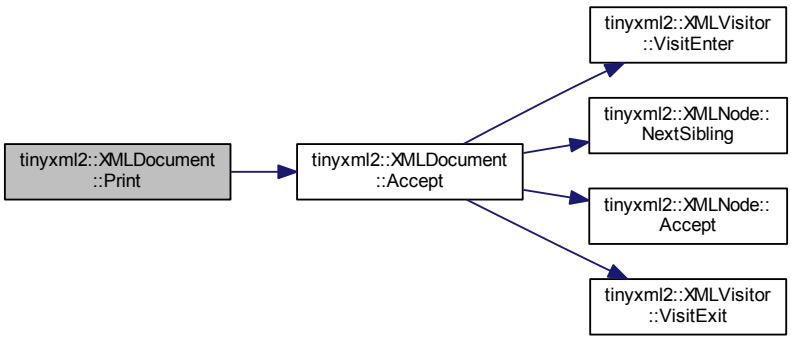
```
XMLPrinter printer( fp );
doc.Print( &printer );
```

Or you can use a printer to print to memory:

```
XMLPrinter printer;
doc.Print( &printer );
// printer.CStr() has a const char* to the XML
```

Definition at line 2041 of file tinyxml2.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

3.23.2.13 RootElement()

```
XMLElement* tinyxml2::XMLDocument::RootElement ( )  [inline]
```

Return the root element of DOM. Equivalent to FirstChildElement(). To get the first node, use FirstChild().

Definition at line 1605 of file tinyxml2.h.

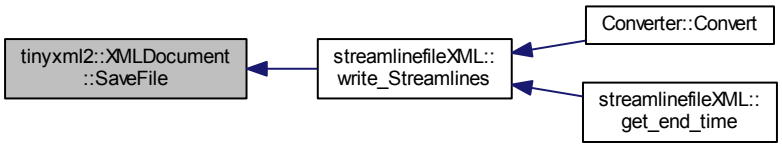
3.23.2.14 SaveFile() [1/2]

```
XMLError tinyxml2::XMLDocument::SaveFile (
            const char * filename,
            bool compact = false )
```

Save the XML file to disk. Returns XML_NO_ERROR (0) on success, or an errorID.

Definition at line 1986 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.23.2.15 SaveFile() [2/2]

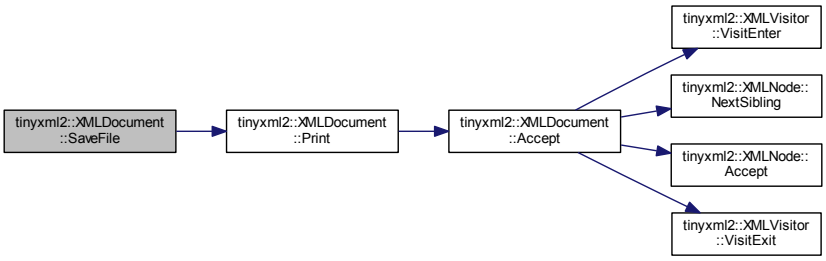
```
XMLError tinyxml2::XMLDocument::SaveFile (
            FILE * fp,
            bool compact = false )
```

Save the XML file to disk. You are responsible for providing and closing the FILE∗.

Returns XML_NO_ERROR (0) on success, or an errorID.

Definition at line 1999 of file tinyxml2.cpp.

Here is the call graph for this function:



3.23.2.16 SetBOM()

```
void tinyxml2::XMLDocument::SetBOM (
            bool useBOM )  [inline]
```

Sets whether to write the BOM when writing the file.

Definition at line 1598 of file tinyxml2.h.

3.23.2.17 ShallowClone()

```
virtual XMLNode* tinyxml2::XMLDocument::ShallowClone (
            XMLDocument * document ) const  [inline], [virtual]
```

Make a copy of this node, but not its children. You may pass in a Document pointer that will be the owner of the new Node. If the 'document' is null, then the node returned will be allocated from the current Document. (this->GetDocument())

Note: if called on a XMLDocument, this will return null.

Implements tinyxml2::XMLNode.

Definition at line 1701 of file tinyxml2.h.

3.23.2.18 ShallowEqual()

```
virtual bool tinyxml2::XMLDocument::ShallowEqual (
            const XMLNode * compare ) const  [inline], [virtual]
```

Test if 2 nodes are the same, but don't test children. The 2 nodes do not need to be in the same Document.

Note: if called on a XMLDocument, this will return false.

Implements tinyxml2::XMLNode.

Definition at line 1704 of file tinyxml2.h.

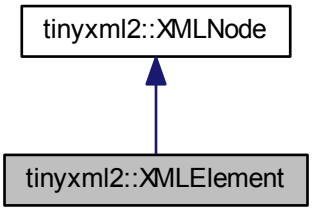
The documentation for this class was generated from the following files:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

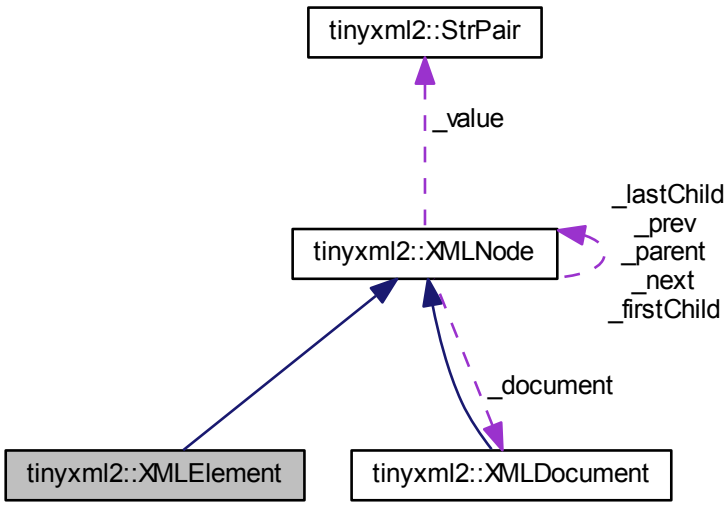
3.24 tinyxml2::XMLElement Class Reference

`#include <tinyxml2.h>`

Inheritance diagram for tinyxml2::XMLElement:



Collaboration diagram for tinyxml2::XMLElement:



Public Types

- enum { **OPEN**, **CLOSED**, **CLOSING** }

Public Member Functions

- const char * Name () const

  Get the name of an element (which is the Value() of the node.)
- void SetName (const char *str, bool staticMem=false)

  Set the name of the element.
- virtual XMLElement * ToElement ()

  Safely cast to an Element, or null.
- virtual const XMLElement * **ToElement** () const
- virtual bool Accept (XMLVisitor *visitor) const
- const char * Attribute (const char *name, const char *value=0) const
- int IntAttribute (const char *name) const
- unsigned UnsignedAttribute (const char *name) const

  See IntAttribute()
- bool BoolAttribute (const char *name) const

  See IntAttribute()
- double DoubleAttribute (const char *name) const

  See IntAttribute()
- float FloatAttribute (const char *name) const

  See IntAttribute()
- XMLError QueryIntAttribute (const char *name, int *value) const
- XMLError QueryUnsignedAttribute (const char *name, unsigned int *value) const

  See QueryIntAttribute()
- XMLError QueryBoolAttribute (const char *name, bool *value) const

  See QueryIntAttribute()
- XMLError QueryDoubleAttribute (const char *name, double *value) const

  See QueryIntAttribute()
- XMLError QueryFloatAttribute (const char *name, float *value) const

  See QueryIntAttribute()
- int QueryAttribute (const char *name, int *value) const
- int **QueryAttribute** (const char *name, unsigned int *value) const
- int **QueryAttribute** (const char *name, bool *value) const
- int **QueryAttribute** (const char *name, double *value) const
- int **QueryAttribute** (const char *name, float *value) const
- void SetAttribute (const char *name, const char *value)

  Sets the named attribute to value.
- void SetAttribute (const char *name, int value)

  Sets the named attribute to value.
- void SetAttribute (const char *name, unsigned value)

  Sets the named attribute to value.
- void SetAttribute (const char *name, bool value)

  Sets the named attribute to value.
- void SetAttribute (const char *name, double value)

  Sets the named attribute to value.
- void SetAttribute (const char *name, float value)

  Sets the named attribute to value.
- void DeleteAttribute (const char *name)
- const XMLAttribute * FirstAttribute () const

  Return the first attribute in the list.
- const XMLAttribute * FindAttribute (const char *name) const

  Query a specific attribute in the list.
- const char * GetText () const

- void SetText (const char *inText)
- void SetText (int value)

  Convenience method for setting text inside an element. See SetText() for important limitations.
- void SetText (unsigned value)

  Convenience method for setting text inside an element. See SetText() for important limitations.
- void SetText (bool value)

  Convenience method for setting text inside an element. See SetText() for important limitations.
- void SetText (double value)

  Convenience method for setting text inside an element. See SetText() for important limitations.
- void SetText (float value)

  Convenience method for setting text inside an element. See SetText() for important limitations.
- XMLError QueryIntText (int *ival) const
- XMLError QueryUnsignedText (unsigned *uval) const

  See QueryIntText()
- XMLError QueryBoolText (bool *bval) const

  See QueryIntText()
- XMLError QueryDoubleText (double *dval) const

  See QueryIntText()
- XMLError QueryFloatText (float *fval) const

  See QueryIntText()
- int **ClosingType** () const
- virtual XMLNode * ShallowClone (XMLDocument *document) const
- virtual bool ShallowEqual (const XMLNode *compare) const

Protected Member Functions

- char * **ParseDeep** (char *p, StrPair *endTag)

Friends

- class **XMLDocument**

Additional Inherited Members

3.24.1 Detailed Description

The element is a container class. It has a value, the element name, and can contain other elements, text, comments, and unknowns. Elements also contain an arbitrary number of attributes.

Definition at line 1142 of file tinyxml2.h.

3.24.2 Member Function Documentation

3.24.2.1 Accept()

```
bool tinyxml2::XMLElement::Accept (
            XMLVisitor * visitor ) const  [virtual]
```

Accept a hierarchical visit of the nodes in the TinyXML-2 DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the XMLVisitor interface.

This is essentially a SAX interface for TinyXML-2. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML-2 is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- http://www.saxproject.org/
- http://c2.com/cgi/wiki?HierarchicalVisitorPattern

Which are both good references for "visiting".

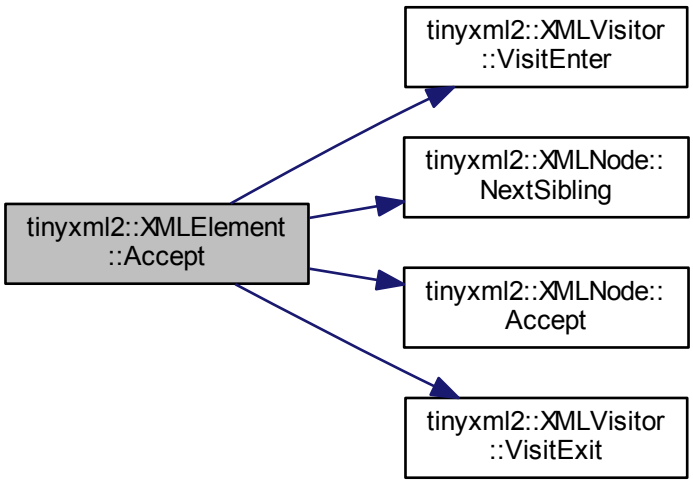
An example of using Accept():

```
XMLPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

Implements tinyxml2::XMLNode.

Definition at line 1729 of file tinyxml2.cpp.

Here is the call graph for this function:



3.24.2.2 Attribute()

```
const char * tinyxml2::XMLElement::Attribute (
            const char * name,
            const char * value = 0 ) const
```

Given an attribute name, Attribute() returns the value for the attribute of that name, or null if none exists. For example:

```
const char* value = ele->Attribute( "foo" );
```

The 'value' parameter is normally null. However, if specified, the attribute will only be returned if the 'name' and 'value' match. This allow you to write code:

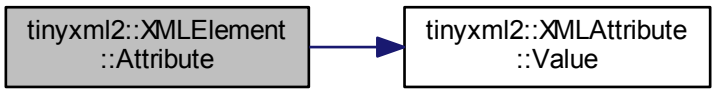
```
if ( ele->Attribute( "foo", "bar" ) ) callFooIsBar();
```

rather than:

```
if ( ele->Attribute( "foo" ) ) {
    if ( strcmp( ele->Attribute( "foo" ), "bar" ) == 0 ) callFooIsBar();
}
```

Definition at line 1404 of file tinyxml2.cpp.

Here is the call graph for this function:



3.24.2.3 DeleteAttribute()

```
void tinyxml2::XMLElement::DeleteAttribute (
            const char * name )
```

Delete an attribute.

Definition at line 1571 of file tinyxml2.cpp.

3.24.2.4 GetText()

```
const char * tinyxml2::XMLElement::GetText ( ) const
```

Convenience function for easy access to the text inside an element. Although easy and concise, GetText() is limited compared to getting the XMLText child and accessing it directly.

If the first child of 'this' is a XMLText, the GetText() returns the character string of the Text node, else null is returned.

This is a convenient method for getting the text of simple contained text:

```
<foo>This is text</foo>
    const char* str = fooElement->GetText();
```

'str' will be a pointer to "This is text".

Note that this function can be misleading. If the element foo was created from this XML:

```
    <foo><b>This is text</b></foo>
```

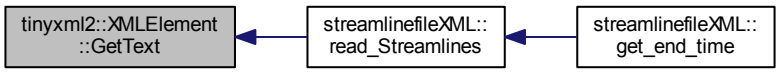
then the value of str would be null. The first child node isn't a text node, it is another element. From this XML:

```
    <foo>This is <b>text</b></foo>
```

GetText() will return "This is ".

Definition at line 1417 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.24.2.5 IntAttribute()

```
int tinyxml2::XMLElement::IntAttribute (
            const char * name ) const  [inline]
```

Given an attribute name, IntAttribute() returns the value of the attribute interpreted as an integer. 0 will be returned if there is an error. For a method with error checking, see QueryIntAttribute()

Definition at line 1193 of file tinyxml2.h.

3.24.2.6 QueryAttribute()

```
int tinyxml2::XMLElement::QueryAttribute (
            const char * name,
            int * value ) const  [inline]
```

Given an attribute name, QueryAttribute() returns XML_NO_ERROR, XML_WRONG_ATTRIBUTE_TYPE if the conversion can't be performed, or XML_NO_ATTRIBUTE if the attribute doesn't exist. It is overloaded for the primitive types, and is a generally more convenient replacement of QueryIntAttribute() and related functions.

If successful, the result of the conversion will be written to 'value'. If not successful, nothing will be written to 'value'. This allows you to provide default value:

```
int value = 10;
QueryAttribute( "foo", &value );        // if "foo" isn't found, value will still be 10
```

Definition at line 1294 of file tinyxml2.h.

3.24.2.7 QueryIntAttribute()

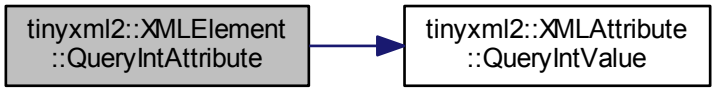
```
XMLError tinyxml2::XMLElement::QueryIntAttribute (
            const char * name,
            int * value ) const  [inline]
```

Given an attribute name, QueryIntAttribute() returns XML_NO_ERROR, XML_WRONG_ATTRIBUTE_TYPE if the conversion can't be performed, or XML_NO_ATTRIBUTE if the attribute doesn't exist. If successful, the result of the conversion will be written to 'value'. If not successful, nothing will be written to 'value'. This allows you to provide default value:

```
int value = 10;
QueryIntAttribute( "foo", &value );     // if "foo" isn't found, value will still be 10
```

Definition at line 1236 of file tinyxml2.h.

Here is the call graph for this function:



3.24.2.8 QueryIntText()

```
XMLError tinyxml2::XMLElement::QueryIntText (
            int * ival ) const
```

Convenience method to query the value of a child text node. This is probably best shown by example. Given you have a document is this form:

```
<point>
    <x>1</x>
    <y>1.4</y>
</point>
```

The QueryIntText() and similar functions provide a safe and easier way to get to the "value" of x and y.

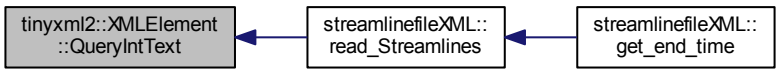
```
int x = 0;
float y = 0;    // types of x and y are contrived for example
const XMLElement* xElement = pointElement->FirstChildElement( "x" );
const XMLElement* yElement = pointElement->FirstChildElement( "y" );
xElement->QueryIntText( &x );
yElement->QueryFloatText( &y );
```

Returns

XML_SUCCESS (0) on success, XML_CAN_NOT_CONVERT_TEXT if the text cannot be converted to the requested type, and XML_NO_TEXT_NODE if there is no child text to query.

Definition at line 1477 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.24.2.9 SetText()

```
void tinyxml2::XMLElement::SetText (
            const char * inText )
```

Convenience function for easy access to the text inside an element. Although easy and concise, SetText() is limited compared to creating an XMLText child and mutating it directly.

If the first child of 'this' is a XMLText, SetText() sets its value to the given string, otherwise it will create a first child that is an XMLText.

This is a convenient method for setting the text of simple contained text:

```
<foo>This is text</foo>
    fooElement->SetText( "Hullaballoo!" );
<foo>Hullaballoo!</foo>
```

Note that this function can be misleading. If the element foo was created from this XML:

```
<foo><b>This is text</b></foo>
```

then it will not change "This is text", but rather prefix it with a text element:

```
<foo>Hullaballoo!<b>This is text</b></foo>
```

For this XML:

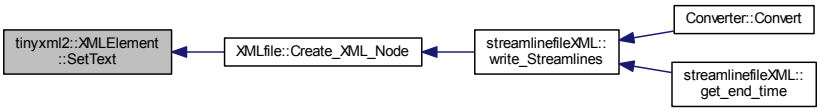
```
<foo />
```

SetText() will generate

```
<foo>Hullaballoo!</foo>
```

Definition at line 1426 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.24.2.10 ShallowClone()

```
XMLNode * tinyxml2::XMLElement::ShallowClone (
            XMLDocument * document ) const  [virtual]
```

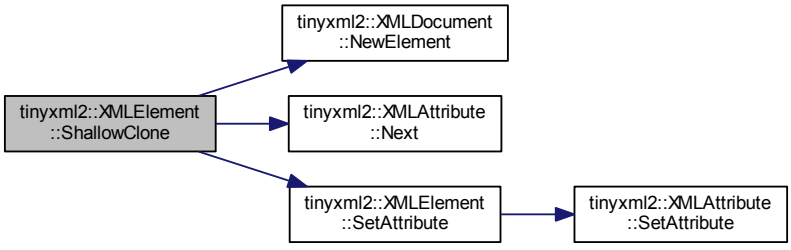
Make a copy of this node, but not its children. You may pass in a Document pointer that will be the owner of the new Node. If the 'document' is null, then the node returned will be allocated from the current Document. (this->GetDocument())

Note: if called on a XMLDocument, this will return null.

Implements tinyxml2::XMLNode.

Definition at line 1690 of file tinyxml2.cpp.

Here is the call graph for this function:



3.24.2.11 ShallowEqual()

```
bool tinyxml2::XMLElement::ShallowEqual (
            const XMLNode * compare ) const  [virtual]
```

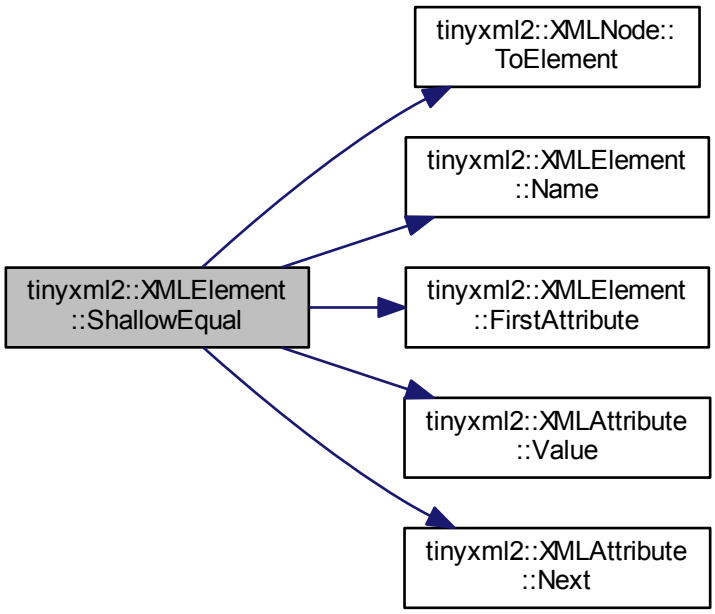
Test if 2 nodes are the same, but don't test children. The 2 nodes do not need to be in the same Document.

Note: if called on a XMLDocument, this will return false.

Implements tinyxml2::XMLNode.

Definition at line 1703 of file tinyxml2.cpp.

Here is the call graph for this function:

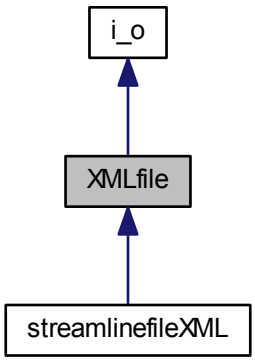


The documentation for this class was generated from the following files:

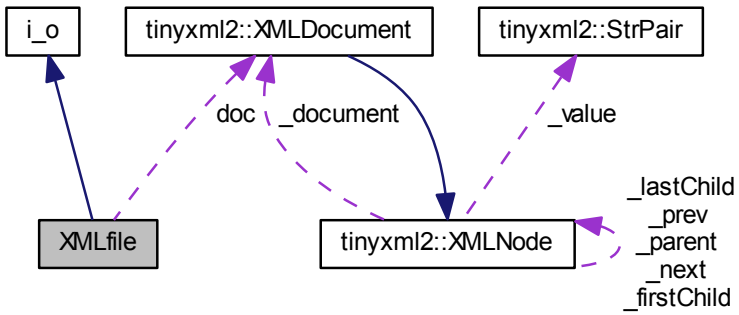
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

3.25 XMLfile Class Reference

Inheritance diagram for XMLfile:



Collaboration diagram for XMLfile:



Protected Member Functions

- void Create_XML_Node (XMLElement *root_e, std::string TAG, std::string text)
- bool check_Tag (XMLElement *e_ptr, std::string TAG)
- void Error_Check (bool status, std::string TAG)

Protected Attributes

- std::string path

  XML file Path.
- XMLDocument doc

  XML file (from tinyxml)

Additional Inherited Members

3.25.1 Detailed Description

Definition at line 9 of file XMLfile.h.

3.25.2 Member Function Documentation

3.25.2.1 check_Tag()

```
bool XMLfile::check_Tag (
            XMLElement * e_ptr,
            std::string TAG )  [protected]
```

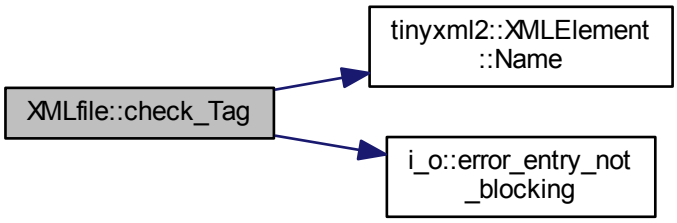
Check if the requested XML TAG is correct

Parameters

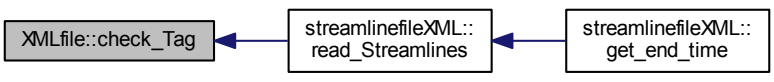
XMLElement	*e_ptr: Pointer to the tinyxml ElementXML to check
std::string	TAG

Definition at line 11 of file XMLfile.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.25.2.2 Create_XML_Node()

```
void XMLfile::Create_XML_Node (
            XMLElement * root_e,
            std::string TAG,
            std::string text )  [protected]
```

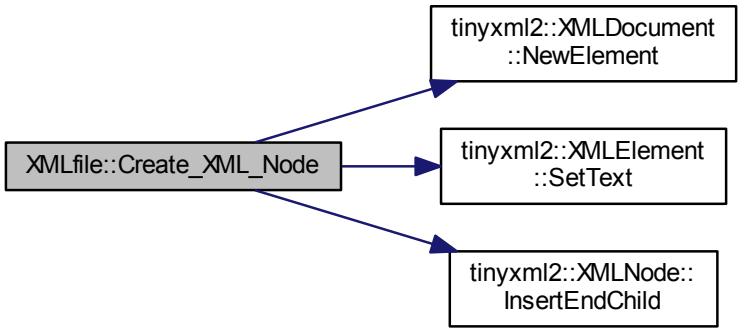
Create Node for the XML File

Parameters

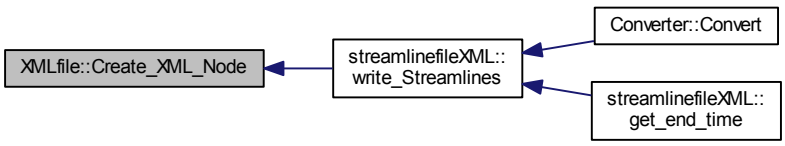
XMLElement	∗cursor_ptr: pointer for expanding the <TAG> sub_tree
std::string	TAG: Name of the node
string	Value of the node

Definition at line 3 of file XMLfile.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



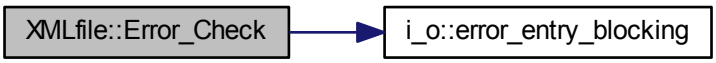
3.25.2.3 Error_Check()

```
void XMLfile::Error_Check (
            bool status,
            std::string TAG )  [protected]
```

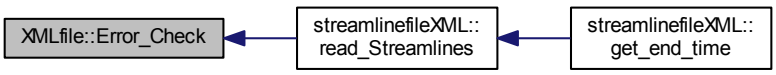
Print out a detalled error and stops the execution bool status: status varible from other processes std::string TAG: XML tag giving problems

Definition at line 29 of file XMLfile.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/XMLfile.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/XMLfile.cpp

3.26 tinyxml2::XMLHandle Class Reference

```
#include <tinyxml2.h>
```

Public Member Functions

- XMLHandle (XMLNode ∗node)

  Create a handle from any node (at any depth of the tree.) This can be a null pointer.
- XMLHandle (XMLNode &node)

  Create a handle from a node.
- XMLHandle (const XMLHandle &ref)

  Copy constructor.
- XMLHandle & operator= (const XMLHandle &ref)

  Assignment.
- XMLHandle FirstChild ()

  Get the first child of this handle.
- XMLHandle FirstChildElement (const char ∗name=0)

  Get the first child element of this handle.
- XMLHandle LastChild ()

  Get the last child of this handle.
- XMLHandle LastChildElement (const char ∗name=0)

  Get the last child element of this handle.
- XMLHandle PreviousSibling ()

  Get the previous sibling of this handle.
- XMLHandle PreviousSiblingElement (const char ∗name=0)

  Get the previous sibling element of this handle.
- XMLHandle NextSibling ()

  Get the next sibling of this handle.
- XMLHandle NextSiblingElement (const char ∗name=0)

  Get the next sibling element of this handle.
- XMLNode ∗ ToNode ()

  Safe cast to XMLNode. This can return null.
- XMLElement ∗ ToElement ()

  Safe cast to XMLElement. This can return null.
- XMLText ∗ ToText ()

  Safe cast to XMLText. This can return null.
- XMLUnknown ∗ ToUnknown ()

  Safe cast to XMLUnknown. This can return null.
- XMLDeclaration ∗ ToDeclaration ()

  Safe cast to XMLDeclaration. This can return null.

3.26.1 Detailed Description

A XMLHandle is a class that wraps a node pointer with null checks; this is an incredibly useful thing. Note that XMLHandle is not part of the TinyXML-2 DOM structure. It is a separate utility class.

Take an example:

```
<Document>
    <Element attributeA = "valueA">
        <Child attributeB = "value1" />
        <Child attributeB = "value2" />
    </Element>
</Document>
```

Assuming you want the value of "attributeB" in the 2nd "Child" element, it's very easy to write a *lot* of code that looks like:

```
XMLElement* root = document.FirstChildElement( "Document" );
if ( root )
{
    XMLElement* element = root->FirstChildElement( "Element" );
    if ( element )
    {
        XMLElement* child = element->FirstChildElement( "Child" );
        if ( child )
        {
            XMLElement* child2 = child->NextSiblingElement( "Child" );
            if ( child2 )
            {
                // Finally do something useful.
```

And that doesn't even cover "else" cases. XMLHandle addresses the verbosity of such code. A XMLHandle checks for null pointers so it is perfectly safe and correct to use:

```
XMLHandle docHandle( &document );
XMLElement* child2 = docHandle.FirstChildElement( "Document" ).FirstChildElement( "Element" ).FirstChildElemen
if ( child2 )
{
    // do something useful
```

Which is MUCH more concise and useful.

It is also safe to copy handles - internally they are nothing more than node pointers.

```
XMLHandle handleCopy = handle;
```

See also XMLConstHandle, which is the same as XMLHandle, but operates on const objects.

Definition at line 1786 of file tinyxml2.h.

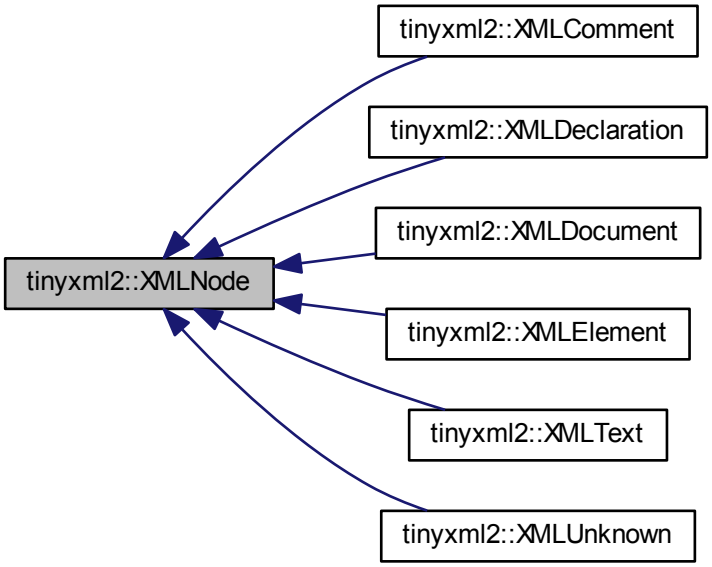
The documentation for this class was generated from the following file:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h

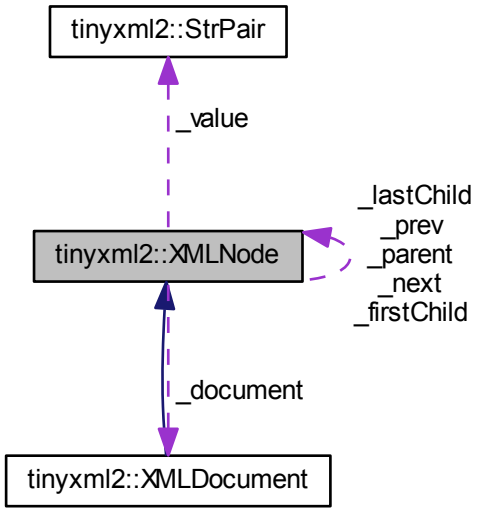
3.27 tinyxml2::XMLNode Class Reference

```
#include <tinyxml2.h>
```

Inheritance diagram for tinyxml2::XMLNode:



Collaboration diagram for tinyxml2::XMLNode:



Public Member Functions

- const XMLDocument * GetDocument () const

Get the XMLDocument that owns this XMLNode.

- XMLDocument * GetDocument ()

  Get the XMLDocument that owns this XMLNode.
- virtual XMLElement * ToElement ()

  Safely cast to an Element, or null.
- virtual XMLText * ToText ()

  Safely cast to Text, or null.
- virtual XMLComment * ToComment ()

  Safely cast to a Comment, or null.
- virtual XMLDocument * ToDocument ()

  Safely cast to a Document, or null.
- virtual XMLDeclaration * ToDeclaration ()

  Safely cast to a Declaration, or null.
- virtual XMLUnknown * ToUnknown ()

  Safely cast to an Unknown, or null.
- virtual const XMLElement * **ToElement** () const
- virtual const XMLText * **ToText** () const
- virtual const XMLComment * **ToComment** () const
- virtual const XMLDocument * **ToDocument** () const
- virtual const XMLDeclaration * **ToDeclaration** () const
- virtual const XMLUnknown * **ToUnknown** () const
- const char * Value () const
- void SetValue (const char *val, bool staticMem=false)
- const XMLNode * Parent () const

  Get the parent of this node on the DOM.
- XMLNode * **Parent** ()
- bool NoChildren () const

  Returns true if this node has no children.
- const XMLNode * FirstChild () const

  Get the first child node, or null if none exists.
- XMLNode * **FirstChild** ()
- const XMLElement * FirstChildElement (const char *name=0) const
- XMLElement * **FirstChildElement** (const char *name=0)
- const XMLNode * LastChild () const

  Get the last child node, or null if none exists.
- XMLNode * **LastChild** ()
- const XMLElement * LastChildElement (const char *name=0) const
- XMLElement * **LastChildElement** (const char *name=0)
- const XMLNode * PreviousSibling () const

  Get the previous (left) sibling node of this node.
- XMLNode * **PreviousSibling** ()
- const XMLElement * PreviousSiblingElement (const char *name=0) const

  Get the previous (left) sibling element of this node, with an optionally supplied name.
- XMLElement * **PreviousSiblingElement** (const char *name=0)
- const XMLNode * NextSibling () const

  Get the next (right) sibling node of this node.
- XMLNode * **NextSibling** ()
- const XMLElement * NextSiblingElement (const char *name=0) const

  Get the next (right) sibling element of this node, with an optionally supplied name.
- XMLElement * **NextSiblingElement** (const char *name=0)
- XMLNode * InsertEndChild (XMLNode *addThis)
- XMLNode * **LinkEndChild** (XMLNode *addThis)

- XMLNode ∗ InsertFirstChild (XMLNode ∗addThis)
- XMLNode ∗ InsertAfterChild (XMLNode ∗afterThis, XMLNode ∗addThis)
- void DeleteChildren ()
- void DeleteChild (XMLNode ∗node)
- virtual XMLNode ∗ ShallowClone (XMLDocument ∗document) const =0
- virtual bool ShallowEqual (const XMLNode ∗compare) const =0
- virtual bool Accept (XMLVisitor ∗visitor) const =0

Protected Member Functions

- **XMLNode** (XMLDocument ∗)
- virtual char ∗ **ParseDeep** (char ∗, StrPair ∗)

Protected Attributes

- XMLDocument ∗ **_document**
- XMLNode ∗ **_parent**
- StrPair **_value**
- XMLNode ∗ **_firstChild**
- XMLNode ∗ **_lastChild**
- XMLNode ∗ **_prev**
- XMLNode ∗ **_next**

Friends

- class **XMLDocument**
- class **XMLElement**

3.27.1 Detailed Description

XMLNode is a base class for every object that is in the XML Document Object Model (DOM), except XMLAttributes. Nodes have siblings, a parent, and children which can be navigated. A node is always in a XMLDocument. The type of a XMLNode can be queried, and it can be cast to its more defined type.

A XMLDocument allocates memory for all its Nodes. When the XMLDocument gets deleted, all its Nodes will also be deleted.

```
A Document can contain: Element (container or leaf)
                        Comment (leaf)
                        Unknown (leaf)
                        Declaration( leaf )

An Element can contain: Element (container or leaf)
                        Text    (leaf)
                        Attributes (not on tree)
                        Comment (leaf)
                        Unknown (leaf)
```

Definition at line 613 of file tinyxml2.h.

3.27.2 Member Function Documentation

3.27.2.1 Accept()

```
virtual bool tinyxml2::XMLNode::Accept (
            XMLVisitor * visitor ) const  [pure virtual]
```

Accept a hierarchical visit of the nodes in the TinyXML-2 DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the XMLVisitor interface.

This is essentially a SAX interface for TinyXML-2. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML-2 is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- http://www.saxproject.org/
- http://c2.com/cgi/wiki?HierarchicalVisitorPattern

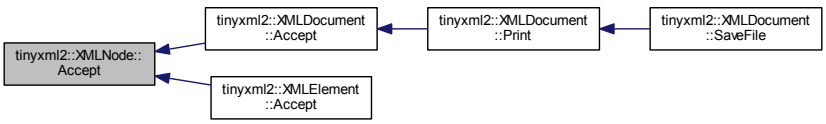
Which are both good references for "visiting".

An example of using Accept():

```
XMLPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

Implemented in tinyxml2::XMLDocument, tinyxml2::XMLElement, tinyxml2::XMLUnknown, tinyxml2::XMLDeclaration, tinyxml2::XMLComment, and tinyxml2::XMLText.

Here is the caller graph for this function:



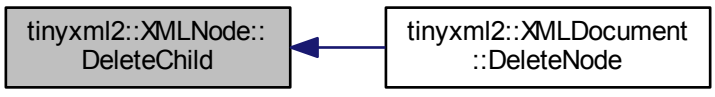
3.27.2.2 DeleteChild()

```
void tinyxml2::XMLNode::DeleteChild (
            XMLNode * node )
```

Delete a child of this node.

Definition at line 774 of file tinyxml2.cpp.

Here is the caller graph for this function:



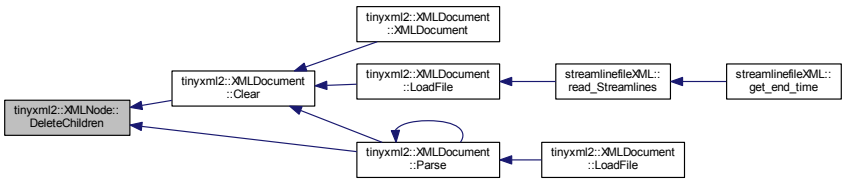
3.27.2.3 DeleteChildren()

```
void tinyxml2::XMLNode::DeleteChildren ( )
```

Delete all the children of this node.

Definition at line 738 of file tinyxml2.cpp.

Here is the caller graph for this function:



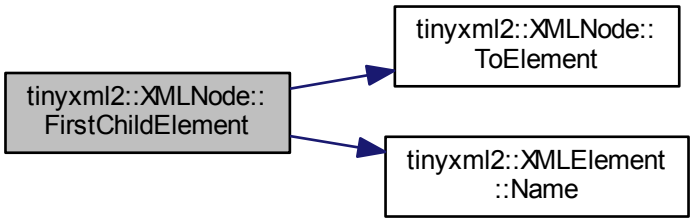
3.27.2.4 FirstChildElement()

```
const XMLElement * tinyxml2::XMLNode::FirstChildElement (
            const char * name = 0 ) const
```

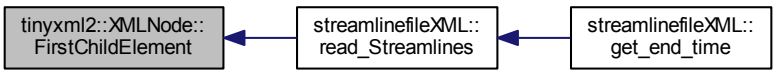
Get the first child element, or optionally the first child element with the specified name.

Definition at line 876 of file tinyxml2.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



3.27.2.5 InsertAfterChild()

```
XMLNode * tinyxml2::XMLNode::InsertAfterChild (
            XMLNode * afterThis,
            XMLNode * addThis )
```

Add a node after the specified child node. If the child node is already part of the document, it is moved from its old location to the new location. Returns the addThis argument or 0 if the afterThis node is not a child of this node, or if the node does not belong to the same document.

Definition at line 845 of file tinyxml2.cpp.

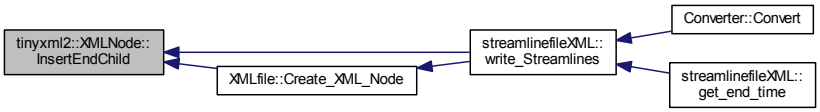
3.27.2.6 InsertEndChild()

```
XMLNode * tinyxml2::XMLNode::InsertEndChild (
            XMLNode * addThis )
```

Add a child node as the last (right) child. If the child node is already part of the document, it is moved from its old location to the new location. Returns the addThis argument or 0 if the node does not belong to the same document.

Definition at line 784 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.27.2.7 InsertFirstChild()

```
XMLNode * tinyxml2::XMLNode::InsertFirstChild (
            XMLNode * addThis )
```

Add a child node as the first (left) child. If the child node is already part of the document, it is moved from its old location to the new location. Returns the addThis argument or 0 if the node does not belong to the same document.

Definition at line 814 of file tinyxml2.cpp.

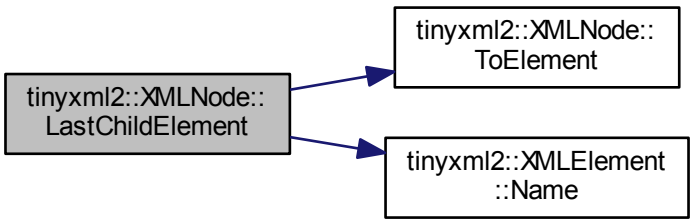
3.27.2.8 LastChildElement()

```
const XMLElement * tinyxml2::XMLNode::LastChildElement (
            const char * name = 0 ) const
```

Get the last child element or optionally the last child element with the specified name.

Definition at line 890 of file tinyxml2.cpp.

Here is the call graph for this function:



3.27.2.9 SetValue()

```
void tinyxml2::XMLNode::SetValue (
            const char * val,
            bool staticMem = false )
```

Set the Value of an XML node.

See also

Value()

Definition at line 727 of file tinyxml2.cpp.

3.27.2.10 ShallowClone()

```
virtual XMLNode* tinyxml2::XMLNode::ShallowClone (
            XMLDocument * document ) const  [pure virtual]
```

Make a copy of this node, but not its children. You may pass in a Document pointer that will be the owner of the new Node. If the 'document' is null, then the node returned will be allocated from the current Document. (this->GetDocument())

Note: if called on a XMLDocument, this will return null.

Implemented in tinyxml2::XMLDocument, tinyxml2::XMLElement, tinyxml2::XMLUnknown, tinyxml2::XMLDeclaration, tinyxml2::XMLComment, and tinyxml2::XMLText.

3.27.2.11 ShallowEqual()

```
virtual bool tinyxml2::XMLNode::ShallowEqual (
            const XMLNode * compare ) const  [pure virtual]
```

Test if 2 nodes are the same, but don't test children. The 2 nodes do not need to be in the same Document.

Note: if called on a XMLDocument, this will return false.

Implemented in tinyxml2::XMLDocument, tinyxml2::XMLElement, tinyxml2::XMLUnknown, tinyxml2::XML↩ Declaration, tinyxml2::XMLComment, and tinyxml2::XMLText.

3.27.2.12 Value()

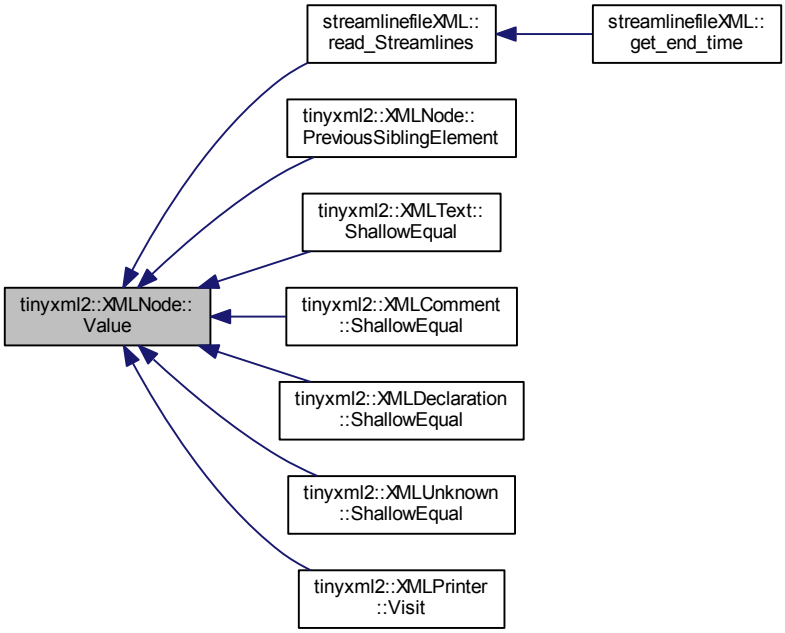
```
const char * tinyxml2::XMLNode::Value ( ) const
```

The meaning of 'value' changes for the specific type.

```
Document:   empty (NULL is returned, not an empty string)
Element:    name of the element
Comment:    the comment text
Unknown:    the tag contents
Text:       the text string
```

Definition at line 719 of file tinyxml2.cpp.

Here is the caller graph for this function:



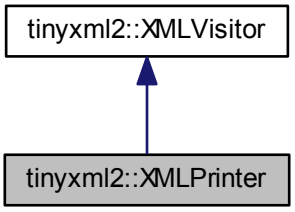
The documentation for this class was generated from the following files:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

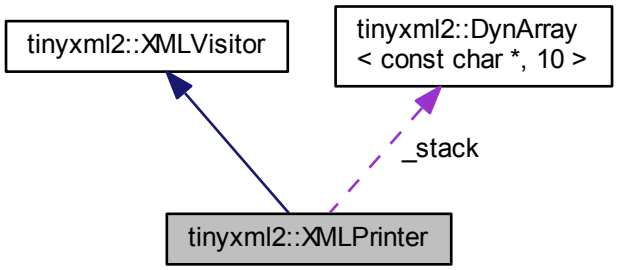
3.28 tinyxml2::XMLPrinter Class Reference

`#include <tinyxml2.h>`

Inheritance diagram for tinyxml2::XMLPrinter:



Collaboration diagram for tinyxml2::XMLPrinter:



Public Member Functions

- XMLPrinter (FILE *file=0, bool compact=false, int depth=0)
- void PushHeader (bool writeBOM, bool writeDeclaration)
- void OpenElement (const char *name, bool compactMode=false)
- void PushAttribute (const char *name, const char *value)

  If streaming, add an attribute to an open element.
- void **PushAttribute** (const char *name, int value)
- void **PushAttribute** (const char *name, unsigned value)
- void **PushAttribute** (const char *name, bool value)
- void **PushAttribute** (const char *name, double value)
- virtual void CloseElement (bool compactMode=false)

  If streaming, close the Element.
- void PushText (const char *text, bool cdata=false)

Add a text node.

- void PushText (int value)

  Add a text node from an integer.
- void PushText (unsigned value)

  Add a text node from an unsigned.
- void PushText (bool value)

  Add a text node from a bool.
- void PushText (float value)

  Add a text node from a float.
- void PushText (double value)

  Add a text node from a double.
- void PushComment (const char *comment)

  Add a comment.
- void **PushDeclaration** (const char *value)
- void **PushUnknown** (const char *value)
- virtual bool VisitEnter (const XMLDocument &)

  Visit a document.
- virtual bool VisitExit (const XMLDocument &)

  Visit a document.
- virtual bool VisitEnter (const XMLElement &element, const XMLAttribute *attribute)

  Visit an element.
- virtual bool VisitExit (const XMLElement &element)

  Visit an element.
- virtual bool Visit (const XMLText &text)

  Visit a text node.
- virtual bool Visit (const XMLComment &comment)

  Visit a comment node.
- virtual bool Visit (const XMLDeclaration &declaration)

  Visit a declaration.
- virtual bool Visit (const XMLUnknown &unknown)

  Visit an unknown node.
- const char * CStr () const
- int CStrSize () const
- void ClearBuffer ()

Protected Member Functions

- virtual bool **CompactMode** (const XMLElement &)
- virtual void PrintSpace (int depth)
- void **Print** (const char *format,...)
- void **SealElementIfJustOpened** ()

Protected Attributes

- bool **_elementJustOpened**
- DynArray< const char *, 10 > **_stack**

3.28.1 Detailed Description

Printing functionality. The XMLPrinter gives you more options than the XMLDocument::Print() method.

It can:

1. Print to memory.
2. Print to a file you provide.
3. Print XML without a XMLDocument.

Print to Memory

```
XMLPrinter printer;
doc.Print( &printer );
SomeFunction( printer.CStr() );
```

Print to a File

You provide the file pointer.

```
XMLPrinter printer( fp );
doc.Print( &printer );
```

Print without a XMLDocument

When loading, an XML parser is very useful. However, sometimes when saving, it just gets in the way. The code is often set up for streaming, and constructing the DOM is just overhead.

The Printer supports the streaming case. The following code prints out a trivially simple XML file without ever creating an XML document.

```
XMLPrinter printer( fp );
printer.OpenElement( "foo" );
printer.PushAttribute( "foo", "bar" );
printer.CloseElement();
```

Definition at line 1977 of file tinyxml2.h.

3.28.2 Constructor & Destructor Documentation

3.28.2.1 XMLPrinter()

```
tinyxml2::XMLPrinter::XMLPrinter (
            FILE * file = 0,
            bool compact = false,
            int depth = 0 )
```

Construct the printer. If the FILE* is specified, this will print to the FILE. Else it will print to memory, and the result is available in CStr(). If 'compact' is set to true, then output is created with only required whitespace and newlines.

Definition at line 2105 of file tinyxml2.cpp.

3.28.3 Member Function Documentation

3.28.3.1 ClearBuffer()

```
void tinyxml2::XMLPrinter::ClearBuffer ( )  [inline]
```

If in print to memory mode, reset the buffer to the beginning.

Definition at line 2055 of file tinyxml2.h.

3.28.3.2 CStr()

```
const char* tinyxml2::XMLPrinter::CStr ( ) const  [inline]
```

If in print to memory mode, return a pointer to the XML file in memory.

Definition at line 2040 of file tinyxml2.h.

3.28.3.3 CStrSize()

```
int tinyxml2::XMLPrinter::CStrSize ( ) const  [inline]
```

If in print to memory mode, return the size of the XML file in memory. (Note the size returned includes the terminating null.)

Definition at line 2048 of file tinyxml2.h.

3.28.3.4 OpenElement()

```
void tinyxml2::XMLPrinter::OpenElement (
            const char * name,
            bool compactMode = false )
```

If streaming, start writing an element. The element must be closed with CloseElement()

Definition at line 2222 of file tinyxml2.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

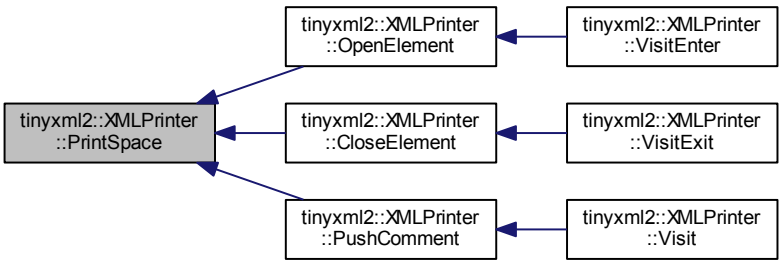
3.28.3.5 PrintSpace()

```
void tinyxml2::XMLPrinter::PrintSpace (
            int depth )  [protected], [virtual]
```

Prints out the space before an element. You may override to change the space and tabs used. A PrintSpace() override should call Print().

Definition at line 2152 of file tinyxml2.cpp.

Here is the caller graph for this function:



3.28.3.6 PushHeader()

```
void tinyxml2::XMLPrinter::PushHeader (
            bool writeBOM,
            bool writeDeclaration )
```

If streaming, write the BOM and declaration.

Definition at line 2210 of file tinyxml2.cpp.

Here is the caller graph for this function:

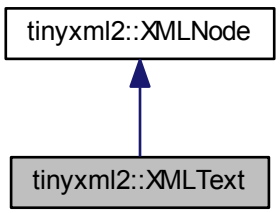
The documentation for this class was generated from the following files:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

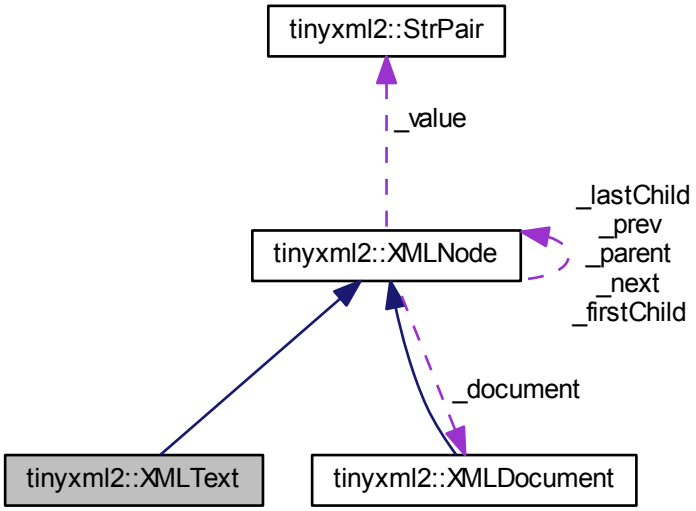
3.29 tinyxml2::XMLText Class Reference

```
#include <tinyxml2.h>
```

Inheritance diagram for tinyxml2::XMLText:



Collaboration diagram for tinyxml2::XMLText:



Public Member Functions

- virtual bool Accept (XMLVisitor ∗visitor) const
- virtual XMLText ∗ ToText ()

  Safely cast to Text, or null.
- virtual const XMLText ∗ **ToText** () const
- void SetCData (bool isCData)

  Declare whether this should be CDATA or standard text.
- bool CData () const

  Returns true if this is a CDATA text element.
- virtual XMLNode ∗ ShallowClone (XMLDocument ∗document) const
- virtual bool ShallowEqual (const XMLNode ∗compare) const

Protected Member Functions

- **XMLText** (XMLDocument ∗doc)
- char ∗ **ParseDeep** (char ∗, StrPair ∗endTag)

Friends

- class **XMLDocument**

Additional Inherited Members

3.29.1 Detailed Description

XML text.

Note that a text node can have child element nodes, for example:

```
<root>This is <b>bold</b></root>
```

A text node can have 2 ways to output the next. "normal" output and CDATA. It will default to the mode it was parsed from the XML file and you generally want to leave it alone, but you can change the output mode with SetCData() and query it with CData().

Definition at line 894 of file tinyxml2.h.

3.29.2 Member Function Documentation

3.29.2.1 Accept()

```
bool tinyxml2::XMLText::Accept (
            XMLVisitor * visitor ) const  [virtual]
```

Accept a hierarchical visit of the nodes in the TinyXML-2 DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the XMLVisitor interface.

This is essentially a SAX interface for TinyXML-2. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML-2 is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- http://www.saxproject.org/
- http://c2.com/cgi/wiki?HierarchicalVisitorPattern

Which are both good references for "visiting".

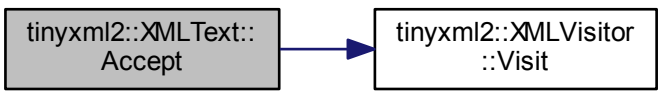
An example of using Accept():

```
XMLPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

Implements tinyxml2::XMLNode.

Definition at line 1085 of file tinyxml2.cpp.

Here is the call graph for this function:



3.29.2.2 ShallowClone()

```
XMLNode * tinyxml2::XMLText::ShallowClone (
            XMLDocument * document ) const  [virtual]
```

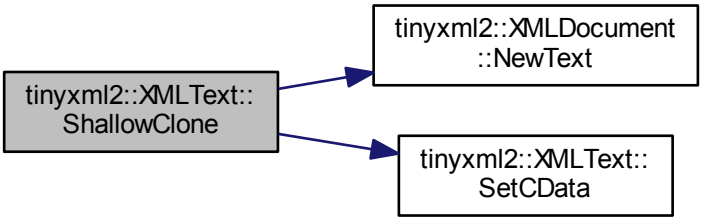
Make a copy of this node, but not its children. You may pass in a Document pointer that will be the owner of the new Node. If the 'document' is null, then the node returned will be allocated from the current Document. (this->GetDocument())

Note: if called on a XMLDocument, this will return null.

Implements tinyxml2::XMLNode.

Definition at line 1067 of file tinyxml2.cpp.

Here is the call graph for this function:



3.29.2.3 ShallowEqual()

```
bool tinyxml2::XMLText::ShallowEqual (
            const XMLNode * compare ) const  [virtual]
```

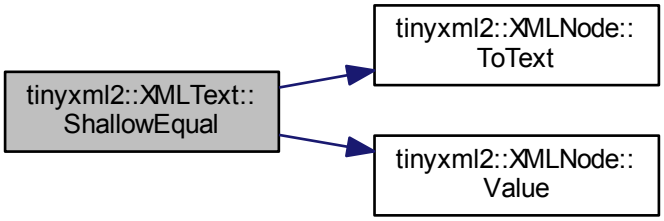
Test if 2 nodes are the same, but don't test children. The 2 nodes do not need to be in the same Document.

Note: if called on a XMLDocument, this will return false.

Implements tinyxml2::XMLNode.

Definition at line 1078 of file tinyxml2.cpp.

Here is the call graph for this function:



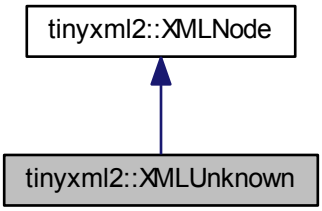
The documentation for this class was generated from the following files:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

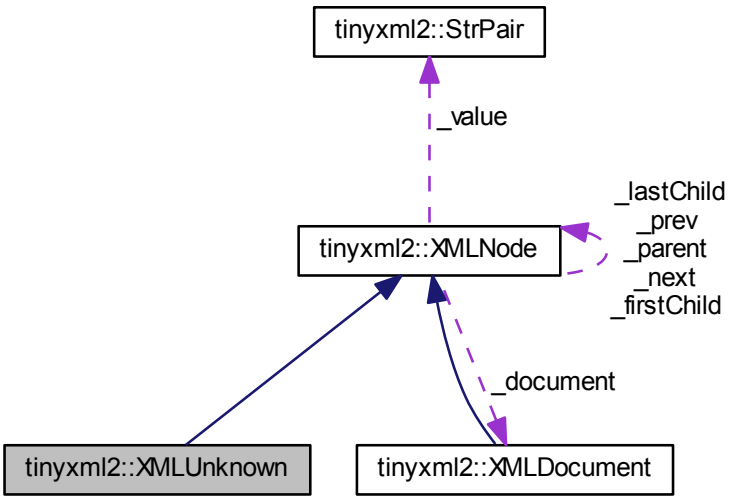
3.30 tinyxml2::XMLUnknown Class Reference

`#include <tinyxml2.h>`

Inheritance diagram for tinyxml2::XMLUnknown:



Collaboration diagram for tinyxml2::XMLUnknown:



Public Member Functions

- virtual XMLUnknown * ToUnknown ()

Safely cast to an Unknown, or null.

- virtual const XMLUnknown * **ToUnknown** () const
- virtual bool Accept (XMLVisitor *visitor) const
- virtual XMLNode * ShallowClone (XMLDocument *document) const
- virtual bool ShallowEqual (const XMLNode *compare) const

Protected Member Functions

- **XMLUnknown** (XMLDocument *doc)
- char * **ParseDeep** (char *, StrPair *endTag)

Friends

- class **XMLDocument**

Additional Inherited Members

3.30.1 Detailed Description

Any tag that TinyXML-2 doesn't recognize is saved as an unknown. It is a tag of text, but should not be modified. It will be written back to the XML, unchanged, when the file is saved.

DTD tags get thrown into XMLUnknowns.

Definition at line 1008 of file tinyxml2.h.

3.30.2 Member Function Documentation

3.30.2.1 Accept()

```
bool tinyxml2::XMLUnknown::Accept (
            XMLVisitor * visitor ) const  [virtual]
```

Accept a hierarchical visit of the nodes in the TinyXML-2 DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the XMLVisitor interface.

This is essentially a SAX interface for TinyXML-2. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML-2 is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- http://www.saxproject.org/
- http://c2.com/cgi/wiki?HierarchicalVisitorPattern

Which are both good references for "visiting".

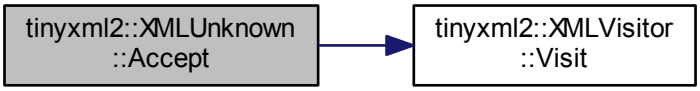
An example of using Accept():

```
XMLPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

Implements tinyxml2::XMLNode.

Definition at line 1234 of file tinyxml2.cpp.

Here is the call graph for this function:



3.30.2.2 ShallowClone()

```
XMLNode * tinyxml2::XMLUnknown::ShallowClone (
            XMLDocument * document ) const  [virtual]
```

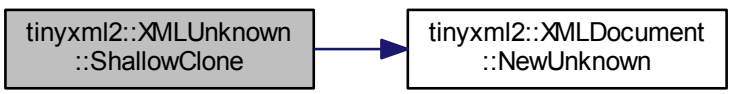
Make a copy of this node, but not its children. You may pass in a Document pointer that will be the owner of the new Node. If the 'document' is null, then the node returned will be allocated from the current Document. (this->GetDocument())

Note: if called on a XMLDocument, this will return null.

Implements tinyxml2::XMLNode.

Definition at line 1216 of file tinyxml2.cpp.

Here is the call graph for this function:



3.30.2.3 ShallowEqual()

```
bool tinyxml2::XMLUnknown::ShallowEqual (
            const XMLNode * compare ) const  [virtual]
```

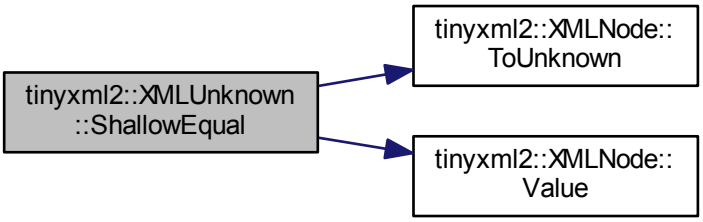
Test if 2 nodes are the same, but don't test children. The 2 nodes do not need to be in the same Document.

Note: if called on a XMLDocument, this will return false.

Implements tinyxml2::XMLNode.

Definition at line 1226 of file tinyxml2.cpp.

Here is the call graph for this function:



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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

3.31 tinyxml2::XMLUtil Class Reference

Static Public Member Functions

- static const char * **SkipWhiteSpace** (const char *p)
- static char * **SkipWhiteSpace** (char *p)
- static bool **IsWhiteSpace** (char p)
- static bool **IsNameStartChar** (unsigned char ch)
- static bool **IsNameChar** (unsigned char ch)
- static bool **StringEqual** (const char *p, const char *q, int nChar=INT_MAX)
- static bool **IsUTF8Continuation** (char p)
- static const char * **ReadBOM** (const char *p, bool *hasBOM)
- static const char * **GetCharacterRef** (const char *p, char *value, int *length)
- static void **ConvertUTF32ToUTF8** (unsigned long input, char *output, int *length)
- static void **ToStr** (int v, char *buffer, int bufferSize)
- static void **ToStr** (unsigned v, char *buffer, int bufferSize)
- static void **ToStr** (bool v, char *buffer, int bufferSize)
- static void **ToStr** (float v, char *buffer, int bufferSize)
- static void **ToStr** (double v, char *buffer, int bufferSize)
- static bool **ToInt** (const char *str, int *value)
- static bool **ToUnsigned** (const char *str, unsigned *value)
- static bool **ToBool** (const char *str, bool *value)
- static bool **ToFloat** (const char *str, float *value)
- static bool **ToDouble** (const char *str, double *value)

3.31.1 Detailed Description

Definition at line 516 of file tinyxml2.h.

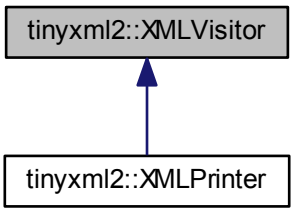
The documentation for this class was generated from the following files:

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.cpp

3.32 tinyxml2::XMLVisitor Class Reference

```
#include <tinyxml2.h>
```

Inheritance diagram for tinyxml2::XMLVisitor:



Public Member Functions

- virtual bool VisitEnter (const XMLDocument &)
  Visit a document.
- virtual bool VisitExit (const XMLDocument &)
  Visit a document.
- virtual bool VisitEnter (const XMLElement &, const XMLAttribute ∗)
  Visit an element.
- virtual bool VisitExit (const XMLElement &)
  Visit an element.
- virtual bool Visit (const XMLDeclaration &)
  Visit a declaration.
- virtual bool Visit (const XMLText &)
  Visit a text node.
- virtual bool Visit (const XMLComment &)
  Visit a comment node.
- virtual bool Visit (const XMLUnknown &)
  Visit an unknown node.

3.32.1 Detailed Description

Implements the interface to the "Visitor pattern" (see the Accept() method.) If you call the Accept() method, it requires being passed a XMLVisitor class to handle callbacks. For nodes that contain other nodes (Document, Element) you will get called with a VisitEnter/VisitExit pair. Nodes that are always leafs are simply called with Visit().

If you return 'true' from a Visit method, recursive parsing will continue. If you return false, **no children of this node or its siblings** will be visited.

All flavors of Visit methods have a default implementation that returns 'true' (continue visiting). You need to only override methods that are interesting to you.

Generally Accept() is called on the XMLDocument, although all nodes support visiting.

You should never change the document from a callback.

See also

XMLNode::Accept()

Definition at line 444 of file tinyxml2.h.

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/tinyxml2/tinyxml2.h

3.33 XY_parser Class Reference

Public Member Functions

- XY_parser ()
  Object blank constructor.
- void parse (std::string filename, Raw_data &parsed_data)
- void map_init ()
  Inititialize support data for the file parsing.

3.33.1 Detailed Description

Definition at line 13 of file XY_parser.h.

3.33.2 Member Function Documentation

3.33.2.1 parse()

```
void XY_parser::parse (
            std::string filename,
            Raw_data & parsed_data )
```

Parse the given XY file

Parameters

std::string	filename: file name
raw_data&	parsed_data reference to the raw data extracted from the files

Definition at line 41 of file XY_parser.cpp.

Here is the caller graph for this function:

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

- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/XY_parser.h
- C:/Users/Francesco/[WS]/NanoDome/Code/Fluent_Link/fluent_link_deliverable_code/XY_parser.cpp

Index