

Optim\_VSTRAP

Generated by Doxygen 1.8.13



# Contents



# Chapter 1

## Deprecated List

**Member [TiXmlHandle::Element](#) () const**

use [ToElement](#). Return the handle as a [TiXmlElement](#). This may return null.

**Member [TiXmlHandle::Node](#) () const**

use [ToNode](#). Return the handle as a [TiXmlNode](#). This may return null.

**Member [TiXmlHandle::Text](#) () const**

use [ToText\(\)](#) Return the handle as a [TiXmlText](#). This may return null.

**Member [TiXmlHandle::Unknown](#) () const**

use [ToUnknown\(\)](#) Return the handle as a [TiXmlUnknown](#). This may return null.



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

abstract_controller . . . . .	??
desired_trajectory_controller . . . . .	??
equation_solving_controller . . . . .	??
gradient_calculator . . . . .	??
input . . . . .	??
objective_calculator . . . . .	??
optim_controller . . . . .	??
output_control_update . . . . .	??
output_diagnostics . . . . .	??
pdf_controller . . . . .	??
stepdirection_controller . . . . .	??
stepsize_controller . . . . .	??
mesh.Cell . . . . .	??
control_field_class.Control_field . . . . .	??
coordinate_phase_space_time . . . . .	??
data_provider . . . . .	??
std::hash< coordinate_phase_space_time > . . . . .	??
mesh.Mesh . . . . .	??
mesh.Node . . . . .	??
particle . . . . .	??
TestCase	
mesh.CellTest . . . . .	??
mesh.MeshTest . . . . .	??
TiXmlAttributeSet . . . . .	??
TiXmlBase . . . . .	??
TiXmlAttribute . . . . .	??
TiXmlNode . . . . .	??
TiXmlComment . . . . .	??
TiXmlDeclaration . . . . .	??
TiXmlDocument . . . . .	??
TiXmlElement . . . . .	??
TiXmlText . . . . .	??
TiXmlUnknown . . . . .	??
TiXmlCursor . . . . .	??
TiXmlHandle . . . . .	??

TiXmlParsingData . . . . .	??
TiXmlString . . . . .	??
TiXmlOutStream . . . . .	??
TiXmlVisitor . . . . .	??
TiXmlPrinter . . . . .	??
FancyArrowPatch	
control_field_class.Arrow3D . . . . .	??



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">abstract_controller</a>	??
<a href="#">control_field_class.Arrow3D</a>	??
<a href="#">mesh.Cell</a>	??
<a href="#">mesh.CellTest</a>	??
<a href="#">control_field_class.Control_field</a>	??
<a href="#">coordinate_phase_space_time</a>	??
<a href="#">data_provider</a>	??
<a href="#">desired_trajectory_controller</a>	??
<a href="#">equation_solving_controller</a>	??
<a href="#">gradient_calculator</a>	??
<a href="#">std::hash&lt; coordinate_phase_space_time &gt;</a>	??
<a href="#">input</a>	??
<a href="#">mesh.Mesh</a>	??
<a href="#">mesh.MeshTest</a>	??
<a href="#">mesh.Node</a>	??
<a href="#">objective_calculator</a>	??
<a href="#">optim_controller</a>	??
<a href="#">output_control_update</a>	
Offers functions to write the update of the control in a file that is readable by the solver for forward and backward equation	??
<a href="#">output_diagnostics</a>	??
<a href="#">particle</a>	??
<a href="#">pdf_controller</a>	??
<a href="#">stepdirection_controller</a>	??
<a href="#">stepsize_controller</a>	??
<a href="#">TiXmlAttribute</a>	??
<a href="#">TiXmlAttributeSet</a>	??
<a href="#">TiXmlBase</a>	??
<a href="#">TiXmlComment</a>	??
<a href="#">TiXmlCursor</a>	??
<a href="#">TiXmlDeclaration</a>	??
<a href="#">TiXmlDocument</a>	??
<a href="#">TiXmlElement</a>	??
<a href="#">TiXmlHandle</a>	??
<a href="#">TiXmlNode</a>	??

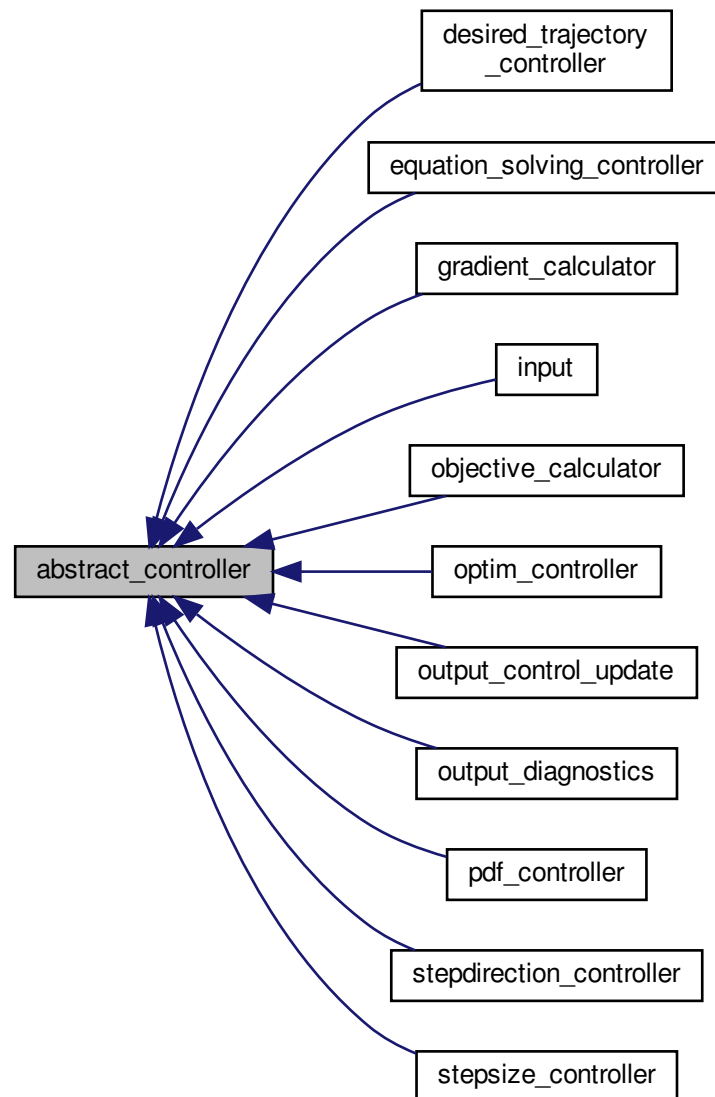
<a href="#">TiXmlOutputStream</a>	??
<a href="#">TiXmlParsingData</a>	??
<a href="#">TiXmlPrinter</a>	??
<a href="#">TiXmlString</a>	??
<a href="#">TiXmlText</a>	??
<a href="#">TiXmlUnknown</a>	??
<a href="#">TiXmlVisitor</a>	??

## **Chapter 4**

### **Class Documentation**

## 4.1 abstract\_controller Class Reference

Inheritance diagram for abstract\_controller:



### Public Member Functions

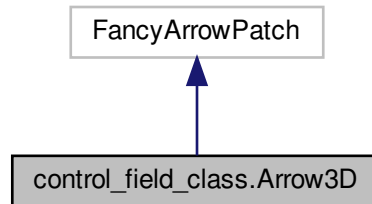
- `data_provider` **getData\_provider\_optim** () const
- void **setData\_provider\_optim** (const `data_provider` &value)

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

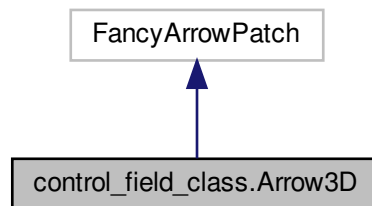
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/abstract_controller.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/abstract_controller.cpp`

## 4.2 control\_field\_class.Arrow3D Class Reference

Inheritance diagram for control\_field\_class.Arrow3D:



Collaboration diagram for control\_field\_class.Arrow3D:



### Public Member Functions

- def **\_\_init\_\_** (self, xs, ys, zs, args, kwargs)
- def **draw** (self, renderer)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/optim-vstrap-toolset/toolset/control\_field\_class.py

## 4.3 mesh.Cell Class Reference

### Public Member Functions

- def **\_\_init\_\_** (self)
- def **set\_nodes** (self, nodes)
- def **calc\_volume** (self, nodes)
- def **calc\_barycenter** (self, nodes)

## Public Attributes

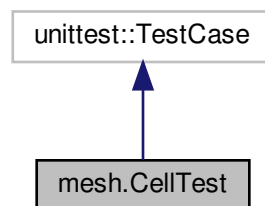
- **id**
- **nodes\_ids**
- **value**
- **volume**
- **type**
- **barycenter**

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

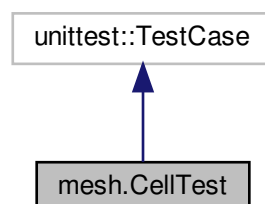
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/optim-vstrap-toolset/toolset/mesh.py

## 4.4 mesh.CellTest Class Reference

Inheritance diagram for mesh.CellTest:



Collaboration diagram for mesh.CellTest:



### Public Member Functions

- def **test\_calc\_volume** (self)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/optim-vstrap-toolset/tests/mesh.py

## 4.5 control\_field\_class.Control\_field Class Reference

### Public Member Functions

- def **\_\_init\_\_** (self)
- def **\_\_str\_\_** (self)
- def **clear** (self)
- def **create\_Lists** (self, controlFile, meshFile)
- def **plot\_Control\_field** (self, nodesMesh, endPoints)

### Public Attributes

- **control**
- **nodesMesh**
- **endPoints**

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/optim-vstrap-toolset/toolset/control\_field\_class.py

## 4.6 coordinate\_phase\_space\_time Class Reference

### Public Member Functions

- **coordinate\_phase\_space\_time** (int cell\_id, int vx, int vy, int vz, int time)
- std::string **toString** () const
- bool **operator==** (const [coordinate\\_phase\\_space\\_time](#) &coordinate) const
- [coordinate\\_phase\\_space\\_time](#) **operator-** (const [coordinate\\_phase\\_space\\_time](#) &coordinate) const
- int **getPx** () const
- void **setPx** (int value)
- int **getPy** () const
- void **setPy** (int value)
- int **getPz** () const
- void **setPz** (int value)
- int **getVx** () const
- void **setVx** (int value)
- int **getVy** () const
- void **setVy** (int value)
- int **getVz** () const
- void **setVz** (int value)
- int **getTime** () const
- void **setTime** (int value)
- int **getCell\_id** () const
- void **setCell\_id** (int value)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/objects/coordinate\_phase\_space\_time.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/objects/coordinate\_phase\_space\_time.cpp

## 4.7 data\_provider Class Reference

### Public Member Functions

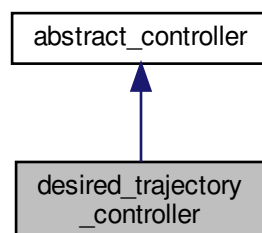
- **data\_provider** (const char \*filename)
- std::map< std::string, std::string > **read\_paths** (const char \*filename)
- std::map< std::string, double > **read\_optimization\_parameters** (const char \*filename)
- std::map< std::string, std::string > **read\_subroutines** (const char \*filename)
- std::map< int, std::vector< double > > **read\_mesh\_barycenters** (const char \*filename)
- std::map< std::string, std::string > **getPaths** () const
- void **setPaths** (const std::map< std::string, std::string > &value)
- std::map< std::string, double > **getOptimizationParameters** () const
- void **setOptimizationParameters** (const std::map< std::string, double > &value)
- std::map< std::string, std::string > **getSubroutines** () const
- void **setSubroutines** (const std::map< std::string, std::string > &value)
- std::map< int, std::vector< double > > **getMesh\_barycenters** () const
- void **setMesh\_barycenters** (const std::map< int, std::vector< double > > &value)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/objects/data\_provider.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/objects/data\_provider.cpp

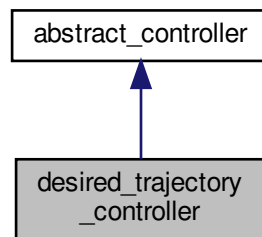
## 4.8 desired\_trajectory\_controller Class Reference

Inheritance diagram for desired\_trajectory\_controller:





Collaboration diagram for desired\_trajectory\_controller:



### Public Member Functions

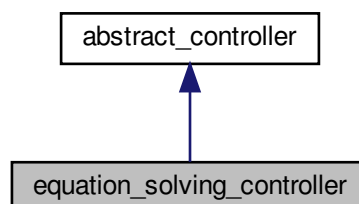
- `std::vector< double > trajectory_desired` (`std::vector< double > barycenter`, unsigned int `l`, unsigned int `m`, unsigned int `n`, unsigned int `o`)
- `std::vector< double > trajectory_desired_harmonic_oscillation` (`std::vector< double > barycenter`, unsigned int `l`, unsigned int `m`, unsigned int `n`, unsigned int `o`)
- `std::vector< double > trajectory_desired_concentrating_mean` (`std::vector< double > barycenter`, unsigned int `l`, unsigned int `m`, unsigned int `n`, unsigned int `o`)
- `std::vector< double > trajectory_desired_shifting_halfbox` (`std::vector< double > barycenter`, unsigned int `l`, unsigned int `m`, unsigned int `n`, unsigned int `o`)
- `std::vector< double > trajectory_desired_concentrating_center` (`std::vector< double > barycenter`, unsigned int `l`, unsigned int `m`, unsigned int `n`, unsigned int `o`)

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

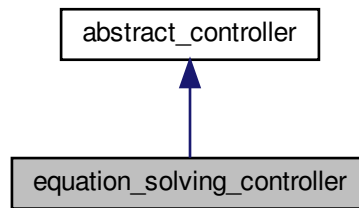
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/desired_trajectory_controller.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/desired_trajectory_controller.cpp`

## 4.9 equation\_solving\_controller Class Reference

Inheritance diagram for equation\_solving\_controller:



Collaboration diagram for `equation_solving_controller`:



### Public Member Functions

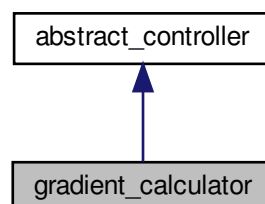
- `int start_solving_forward (std::string start_forward)`
- `int start_solving_backward (std::string start_backward)`
- `arma::mat Laplacian_3D ()`
- `arma::mat Laplacian_Squared_3D ()`

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

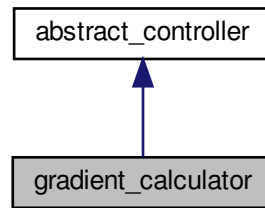
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/equation_solving_controller.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/equation_solving_controller.cpp`

## 4.10 `gradient_calculator` Class Reference

Inheritance diagram for `gradient_calculator`:



Collaboration diagram for gradient\_calculator:



### Public Member Functions

- **gradient\_calculator** (const char \*filename)
- arma::mat **calculateGradient\_forceControl\_space\_L2** (std::vector< std::unordered\_map< [coordinate\\_phase\\_space\\_time](#), double >> forwardPDF\_time, std::vector< std::unordered\_map< [coordinate\\_phase\\_space\\_time](#), double >> backwardPDF\_time, arma::mat control)
- arma::mat **calculateGradient\_forceControl\_space\_Hm** (std::vector< std::unordered\_map< [coordinate\\_phase\\_space\\_time](#), double >> forwardPDF\_time, std::vector< std::unordered\_map< [coordinate\\_phase\\_space\\_time](#), double >> backwardPDF\_time, arma::mat control)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/gradient\_calculator.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/gradient\_calculator.cpp

## 4.11 std::hash< coordinate\_phase\_space\_time > Struct Template Reference

### Public Types

- typedef [coordinate\\_phase\\_space\\_time](#) **argument\_type**
- typedef size\_t **result\_type**

### Public Member Functions

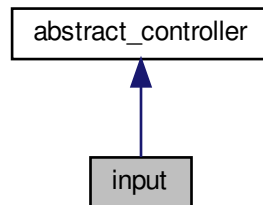
- size\_t **operator()** (const [argument\\_type](#) &x) const

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

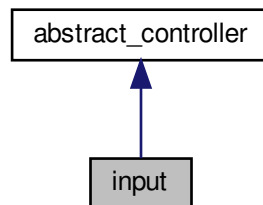
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/objects/coordinate\_phase\_space\_time.h

## 4.12 input Class Reference

Inheritance diagram for input:



Collaboration diagram for input:



### Public Member Functions

- unsigned int **read\_plasma\_state\_forward** (std::vector< std::vector< [particle](#) >> &forwardParticles)
- unsigned int **read\_plasma\_state\_backward** (std::vector< std::vector< [particle](#) >> &backwardParticles)

### Static Public Member Functions

- static std::vector< [particle](#) > **readParticleVector** (std::string filename, std::string delimiter)
- static arma::mat **readControl** (const char \*filename)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/input.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/input.cpp

## 4.13 mesh.Mesh Class Reference

### Public Member Functions

- `def __init__ (self)`
- `def __str__ (self)`
- `def clear (self)`
- `def read_mesh_xml (self, file_name)`
- `def interpolate_cell2node (self)`
- `def read_control_csv (self, file_name)`
- `def read_control_xml (self, file_name)`
- `def write_control_csv (self, file_name)`
- `def write_control_xml (self, file_name)`
- `def write_barycenters_xml (self, file_name)`

### Public Attributes

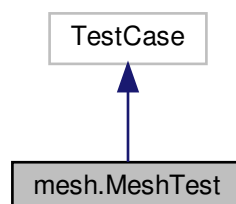
- **cells**
- **nodes**
- **volume**

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

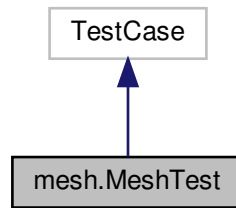
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/optim-vstrap-toolset/toolset/mesh.py`

## 4.14 mesh.MeshTest Class Reference

Inheritance diagram for mesh.MeshTest:



Collaboration diagram for mesh.MeshTest:



### Public Member Functions

- def **test\_read\_mesh\_xml** (self)
- def **test\_read\_control\_csv** (self)
- def **test\_read\_control\_xml** (self)
- def **test\_interpolate\_cell2node** (self)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/optim-vstrap-toolset/tests/mesh.py

## 4.15 mesh.Node Class Reference

### Public Member Functions

- def **\_\_init\_\_** (self, id=0, coord=(0.0, 0.0, 0.0))
- def **get\_position** (self)

### Public Attributes

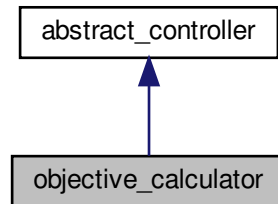
- **id**
- **x\_coord**
- **y\_coord**
- **z\_coord**
- **value**

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

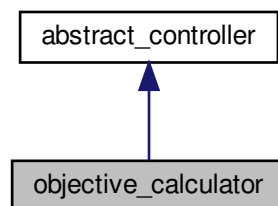
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/optim-vstrap-toolset/toolset/mesh.py

## 4.16 objective\_calculator Class Reference

Inheritance diagram for objective\_calculator:



Collaboration diagram for objective\_calculator:



### Public Member Functions

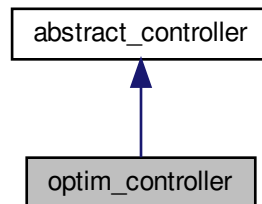
- **objective\_calculator** (const char \*filename)
- double **calculate\_objective\_L2** (std::vector< std::unordered\_map< [coordinate\\_phase\\_space\\_time](#), double >> forwardPDF\_time, arma::mat control)

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

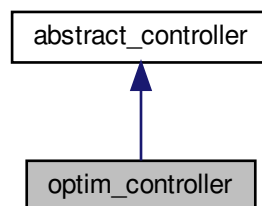
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/objective\_calculator.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/objective\_calculator.cpp

## 4.17 optim\_controller Class Reference

Inheritance diagram for optim\_controller:



Collaboration diagram for optim\_controller:



### Public Member Functions

- int **start\_optimizer** (int argc, const char \*\*argv)

### Static Public Member Functions

- static int [start\\_optimization\\_iteration](#) (const char \*input\_xml\_path)
- static std::unordered\_map< [coordinate\\_phase\\_space\\_time](#), double > **assemblePDF\_thread** (std::vector< std::vector< [particle](#) >> &particles, unsigned int equation\_type, [data\\_provider](#) data\_provider\_)
- static int **check\_input\_py** ([data\\_provider](#) provider, const char \*filePathOptimInput)
- static int **interpolate\_control** ([data\\_provider](#) provider)
- static arma::mat **start\_with\_zero\_control** (const char \*input\_xml\_path)
- static arma::mat **start\_with\_given\_control** (const char \*input\_xml\_path)

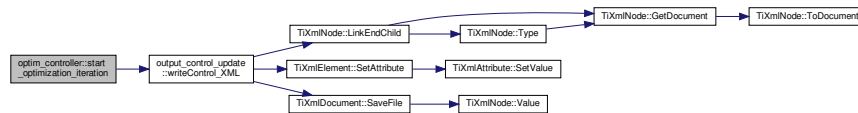
#### 4.17.1 Member Function Documentation



## 4.17.1.1 start\_optimization\_iteration()

```
int optim_controller::start_optimization_iteration (
    const char * input_xml_path ) [static]
```

START OPTIMIZATION ITERATION Here is the call graph for this function:



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

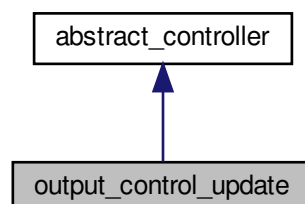
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/controller/optim\_controller.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/controller/optim\_controller.cpp

## 4.18 output\_control\_update Class Reference

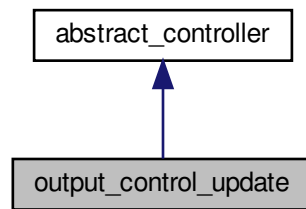
The [output\\_control\\_update](#) class offers functions to write the update of the control in a file that is readable by the solver for forward and backward equation.

```
#include <output_control_update.h>
```

Inheritance diagram for output\_control\_update:



Collaboration diagram for `output_control_update`:



## Public Member Functions

- **`output_control_update`** (`const char *filename`)
- `int writeControl_XML` (`arma::mat control`)  
*writeControl\_XML*

### 4.18.1 Detailed Description

The `output_control_update` class offers functions to write the update of the control in a file that is readable by the solver for forward and backward equation.

### 4.18.2 Member Function Documentation

#### 4.18.2.1 `writeControl_XML()`

```
int output_control_update::writeControl_XML (
    arma::mat control )
```

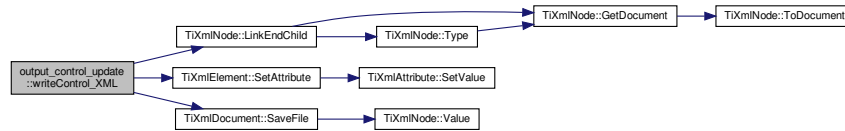
`writeControl_XML`

#### Parameters

<i>control</i>	
----------------	--

## Returns

Here is the call graph for this function:

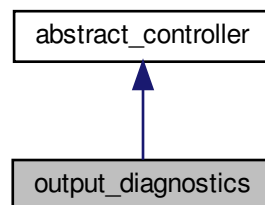


The documentation for this class was generated from the following files:

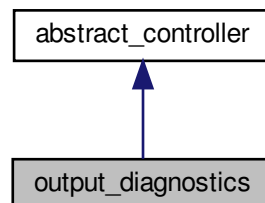
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/output_control_update.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/output_control_update.cpp`

## 4.19 output\_diagnostics Class Reference

Inheritance diagram for `output_diagnostics`:



Collaboration diagram for `output_diagnostics`:



## Public Member Functions

- int **writeGradientToFile** (arma::mat gradient, std::string filename)
- int **writeDoubleToFile** (double value, std::string filename)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/output\_diagnostics.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/output\_diagnostics.cpp

## 4.20 particle Class Reference

### Public Member Functions

- **particle** (double vx, double vy, double vz)
- **particle** (double px, double py, double pz, double vx, double vy, double vz)
- **particle** (double px, double py, double pz, double vx, double vy, double vz, int cell\_id)
- bool **operator==** (const [particle](#) &[particle](#)) const
- double [getVelocityMagnitudeParticle](#) ()  
*getVelocityMagnitudeParticle calculates speed of particles using Euclidean Norm*
- std::string [toString](#) ()  
*toString*
- double **getPx** () const
- void **setPx** (double value)
- double **getPy** () const
- void **setPy** (double value)
- double **getPz** () const
- void **setPz** (double value)
- double **getVx** () const
- void **setVx** (double value)
- double **getVy** () const
- void **setVy** (double value)
- double **getVz** () const
- void **setVz** (double value)
- int **getCell\_id** () const
- void **setCell\_id** (int value)
- double **getWeight** () const
- void **setWeight** (double value)

### 4.20.1 Member Function Documentation

#### 4.20.1.1 [getVelocityMagnitudeParticle\(\)](#)

```
double particle::getVelocityMagnitudeParticle ( )
```

[getVelocityMagnitudeParticle](#) calculates speed of particles using Euclidean Norm

**Returns**

## 4.20.1.2 toString()

```
std::string particle::toString ( )
```

toString

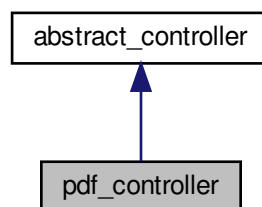
Returns

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

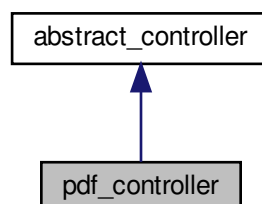
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/objects/particle.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/objects/particle.cpp

## 4.21 pdf\_controller Class Reference

Inheritance diagram for pdf\_controller:



Collaboration diagram for pdf\_controller:



## Public Member Functions

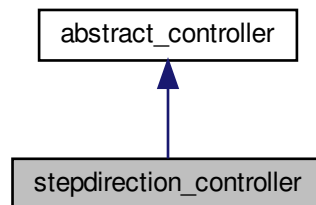
- `std::unordered_map< coordinate\_phase\_space\_time, double > assemblingMultiDim` (`std::vector< std::vector< particle > > &particlesTime`, `unsigned int equationType`)
- `std::vector< std::unordered_map< coordinate\_phase\_space\_time, double > > assemblingMultiDim_parallel` (`std::vector< std::vector< particle > > &particlesTime`, `unsigned int equationType`)
- `std::vector< std::vector< std::vector< std::vector< double > > > > relaxing_GaussSeidel_4D` (`std::vector< std::vector< std::vector< std::vector< double > > > pdf`, `unsigned int numberOfRelaxationSteps`)
- `double calculate_wasserstein_metric` (`std::vector< std::vector< particle > > dist1`, `std::vector< std::vector< particle > > dist2`)
- `double calculate_wasserstein_metric_histogramm` (`std::vector< std::unordered_map< coordinate\_phase\_space\_time, double > > dist1`, `std::vector< std::unordered_map< coordinate\_phase\_space\_time, double > > dist2`)

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

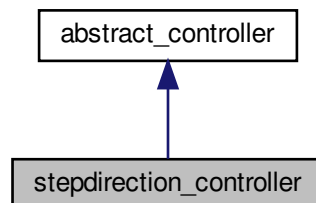
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/pdf_controller.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/controller/pdf_controller.cpp`

## 4.22 stepdirection\_controller Class Reference

Inheritance diagram for `stepdirection_controller`:



Collaboration diagram for `stepdirection_controller`:



## Public Member Functions

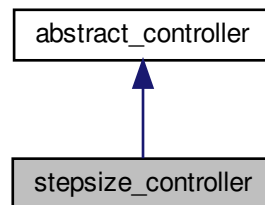
- **stepdirection\_controller** (const char \*filename)
- arma::mat **get\_stepdirection** (arma::mat gradient, arma::mat gradient\_old, arma::mat stepdirectionOld, unsigned int optimization\_iteration)
- arma::mat **fixed\_gradient\_descent** (arma::mat gradient, unsigned int optimization\_iteration)
- arma::mat **ncg\_scheme\_FR** (arma::mat gradient, arma::mat gradient\_old, arma::mat stepdirectionOld, unsigned int optimization\_iteration)
- arma::mat **ncg\_scheme\_PR** (arma::mat gradient, arma::mat gradient\_old, arma::mat stepdirectionOld, unsigned int optimization\_iteration)
- arma::mat **ncg\_scheme\_HZ** (arma::mat gradient, arma::mat gradient\_old, arma::mat stepdirectionOld, unsigned int optimization\_iteration)

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

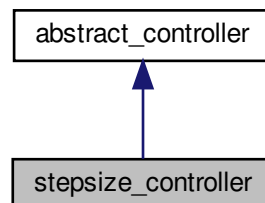
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/stepdirection\_controller.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/stepdirection\_controller.cpp

## 4.23 stepsize\_controller Class Reference

Inheritance diagram for stepsize\_controller:



Collaboration diagram for stepsize\_controller:



## Public Member Functions

- **stepsize\_controller** (const char \*filename)
- int **calculate\_stepsize** (arma::mat &gradient, double J0, arma::mat &control, arma::mat &stepdirection, std::vector< [particle](#) > &inputParticles, double &stepsize0)

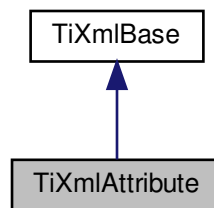
The documentation for this class was generated from the following files:

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/stepsize\_controller.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/optimization/stepsize\_controller.cpp

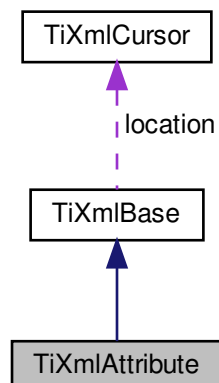
## 4.24 TiXmlAttribute Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlAttribute:



Collaboration diagram for TiXmlAttribute:





## Public Member Functions

- [TiXmlAttribute](#) ()  
*Construct an empty attribute.*
- [TiXmlAttribute](#) (const char \*\_name, const char \*\_value)  
*Construct an attribute with a name and value.*
- const char \* [Name](#) () const  
*Return the name of this attribute.*
- const char \* [Value](#) () const  
*Return the value of this attribute.*
- int [IntValue](#) () const  
*Return the value of this attribute, converted to an integer.*
- double [DoubleValue](#) () const  
*Return the value of this attribute, converted to a double.*
- const TIXML\_STRING & [NameTStr](#) () const
- int [QueryIntValue](#) (int \*\_value) const
- int [QueryDoubleValue](#) (double \*\_value) const  
*QueryDoubleValue examines the value string. See [QueryIntValue\(\)](#).*
- void [SetName](#) (const char \*\_name)  
*Set the name of this attribute.*
- void [SetValue](#) (const char \*\_value)  
*Set the value.*
- void [SetIntValue](#) (int \_value)  
*Set the value from an integer.*
- void [SetDoubleValue](#) (double \_value)  
*Set the value from a double.*
- const [TiXmlAttribute](#) \* [Next](#) () const  
*Get the next sibling attribute in the DOM. Returns null at end.*
- [TiXmlAttribute](#) \* [Next](#) ()
- const [TiXmlAttribute](#) \* [Previous](#) () const  
*Get the previous sibling attribute in the DOM. Returns null at beginning.*
- [TiXmlAttribute](#) \* [Previous](#) ()
- bool **operator==** (const [TiXmlAttribute](#) &rhs) const
- bool **operator<** (const [TiXmlAttribute](#) &rhs) const
- bool **operator>** (const [TiXmlAttribute](#) &rhs) const
- virtual const char \* **Parse** (const char \*p, [TiXmlParsingData](#) \*data, TiXmlEncoding encoding)
- virtual void [Print](#) (FILE \*cfile, int depth) const
- void **Print** (FILE \*cfile, int depth, TIXML\_STRING \*str) const
- void **SetDocument** ([TiXmlDocument](#) \*doc)

## Friends

- class [TiXmlAttributeSet](#)

## Additional Inherited Members

### 4.24.1 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 [TiXmlNode](#)s, since they are not part of the tinyXML document object model. There are other suggested ways to look at this problem.

## 4.24.2 Member Function Documentation

### 4.24.2.1 Print()

```
virtual void TiXmlAttribute::Print (
    FILE * cfile,
    int depth ) const [inline], [virtual]
```

All TinyXml classes can print themselves to a filestream or the string class ([TiXmlString](#) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements [TiXmlBase](#).

### 4.24.2.2 QueryIntValue()

```
int TiXmlAttribute::QueryIntValue (
    int * _value ) const
```

`QueryIntValue` examines the value string. It is an alternative to the [IntValue\(\)](#) method with richer error checking. If the value is an integer, it is stored in 'value' and the call returns `TIXML_SUCCESS`. If it is not an integer, it returns `TIXML_WRONG_TYPE`.

A specialized but useful call. Note that for success it returns 0, which is the opposite of almost all other TinyXml calls.

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

- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.cpp`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp`

## 4.25 TiXmlAttributeSet Class Reference

### Public Member Functions

- void **Add** ([TiXmlAttribute](#) \*attribute)
- void **Remove** ([TiXmlAttribute](#) \*attribute)
- const [TiXmlAttribute](#) \* **First** () const
- [TiXmlAttribute](#) \* **First** ()
- const [TiXmlAttribute](#) \* **Last** () const
- [TiXmlAttribute](#) \* **Last** ()
- [TiXmlAttribute](#) \* **Find** (const char \*\_name) const
- [TiXmlAttribute](#) \* **FindOrCreate** (const char \*\_name)

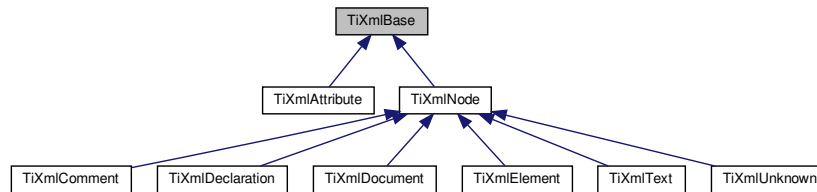
The documentation for this class was generated from the following files:

- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.cpp`

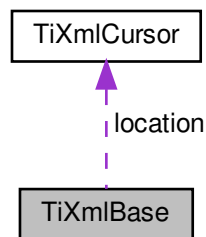
## 4.26 TiXmlBase Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlBase:



Collaboration diagram for TiXmlBase:



### Public Types

- enum {  
 TIXML\_NO\_ERROR = 0, TIXML\_ERROR, TIXML\_ERROR\_OPENING\_FILE, TIXML\_ERROR\_PARSING\_ELEMENT,  
 TIXML\_ERROR\_FAILED\_TO\_READ\_ELEMENT\_NAME, TIXML\_ERROR\_READING\_ELEMENT\_VALUE,  
 TIXML\_ERROR\_READING\_ATTRIBUTES, TIXML\_ERROR\_PARSING\_EMPTY,  
 TIXML\_ERROR\_READING\_END\_TAG, TIXML\_ERROR\_PARSING\_UNKNOWN, TIXML\_ERROR\_PARSING\_COMMENT,  
 TIXML\_ERROR\_PARSING\_DECLARATION,  
 TIXML\_ERROR\_DOCUMENT\_EMPTY, TIXML\_ERROR\_EMBEDDED\_NULL, TIXML\_ERROR\_PARSING\_CDATA,  
 TIXML\_ERROR\_DOCUMENT\_TOP\_ONLY,  
 TIXML\_ERROR\_STRING\_COUNT }

### Public Member Functions

- virtual void [Print](#) (FILE \*cfile, int depth) const =0
- int [Row](#) () const
- int [Column](#) () const

- See [Row\(\)](#)
- void [SetUserData](#) (void \*user)  
*Set a pointer to arbitrary user data.*
- void \* [GetUserData](#) ()  
*Get a pointer to arbitrary user data.*
- const void \* [GetUserData](#) () const  
*Get a pointer to arbitrary user data.*
- virtual const char \* **Parse** (const char \*p, [TiXmlParsingData](#) \*data, TiXmlEncoding encoding)=0

### Static Public Member Functions

- static void [SetCondenseWhiteSpace](#) (bool condense)
- static bool [IsWhiteSpaceCondensed](#) ()  
*Return the current white space setting.*
- static void [EncodeString](#) (const TIXML\_STRING &str, TIXML\_STRING \*out)

### Static Public Attributes

- static const int **utf8ByteTable** [256]

### Static Protected Member Functions

- static const char \* **SkipWhiteSpace** (const char \*, TiXmlEncoding encoding)
- static bool **IsWhiteSpace** (char c)
- static bool **IsWhiteSpace** (int c)
- static const char \* **ReadName** (const char \*p, TIXML\_STRING \*name, TiXmlEncoding encoding)
- static const char \* **ReadText** (const char \*in, TIXML\_STRING \*text, bool ignoreWhiteSpace, const char \*endTag, bool ignoreCase, TiXmlEncoding encoding)
- static const char \* **GetEntity** (const char \*in, char \*value, int \*length, TiXmlEncoding encoding)
- static const char \* **GetChar** (const char \*p, char \*\_value, int \*length, TiXmlEncoding encoding)
- static bool **StringEqual** (const char \*p, const char \*endTag, bool ignoreCase, TiXmlEncoding encoding)
- static int **IsAlpha** (unsigned char anyByte, TiXmlEncoding encoding)
- static int **IsAlphaNum** (unsigned char anyByte, TiXmlEncoding encoding)
- static int **ToLower** (int v, TiXmlEncoding encoding)
- static void **ConvertUTF32ToUTF8** (unsigned long [input](#), char \*output, int \*length)

### Protected Attributes

- [TiXmlCursor](#) **location**
- void \* [userData](#)  
*Field containing a generic user pointer.*

### Static Protected Attributes

- static const char \* **errorString** [TIXML\_ERROR\_STRING\_COUNT]

## Friends

- class **TiXmlNode**
- class **TiXmlElement**
- class **TiXmlDocument**

### 4.26.1 Detailed Description

**TiXmlBase** is a base class for every class in TinyXml. It does little except to establish that TinyXml classes can be printed and provide some utility functions.

In XML, the document and elements can contain other elements and other types of nodes.

```
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)
```

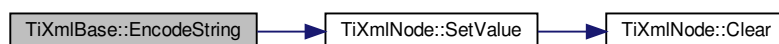
```
A Declaration contains: Attributes (not on tree)
```

### 4.26.2 Member Function Documentation

#### 4.26.2.1 EncodeString()

```
void TiXmlBase::EncodeString (
    const TIXML_STRING & str,
    TIXML_STRING * out ) [static]
```

Expands entities in a string. Note this should not contain the tag's '<', '>', etc, or they will be transformed into entities! Here is the call graph for this function:



#### 4.26.2.2 Print()

```
virtual void TiXmlBase::Print (
    FILE * cfile,
    int depth ) const [pure virtual]
```

All TinyXml classes can print themselves to a filestream or the string class ([TiXmlString](#) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implemented in [TiXmlDocument](#), [TiXmlUnknown](#), [TiXmlDeclaration](#), [TiXmlText](#), [TiXmlComment](#), [TiXmlElement](#), and [TiXmlAttribute](#).

#### 4.26.2.3 Row()

```
int TiXmlBase::Row ( ) const [inline]
```

Return the position, in the original source file, of this node or attribute. The row and column are 1-based. (That is the first row and first column is 1,1). If the returns values are 0 or less, then the parser does not have a row and column value.

Generally, the row and column value will be set when the `TiXmlDocument::Load()`, [TiXmlDocument::LoadFile\(\)](#), or any `TiXmlNode::Parse()` is called. It will NOT be set when the DOM was created from operator`>>`.

The values reflect the initial load. Once the DOM is modified programmatically (by adding or changing nodes and attributes) the new values will NOT update to reflect changes in the document.

There is a minor performance cost to computing the row and column. Computation can be disabled if [TiXmlDocument::SetTabSize\(\)](#) is called with 0 as the value.

See also

[TiXmlDocument::SetTabSize\(\)](#)

#### 4.26.2.4 SetCondenseWhiteSpace()

```
static void TiXmlBase::SetCondenseWhiteSpace (
    bool condense ) [inline], [static]
```

The world does not agree on whether white space should be kept or not. In order to make everyone happy, these global, static functions are provided to set whether or not TinyXml will condense all white space into a single space or not. The default is to condense. Note changing this value is not thread safe.

### 4.26.3 Member Data Documentation

#### 4.26.3.1 errorString

```
const char * TiXmlBase::errorString [static], [protected]
```

**Initial value:**

```
{
    "No error",
    "Error",
    "Failed to open file",
    "Error parsing Element.",
    "Failed to read Element name",
    "Error reading Element value.",
    "Error reading Attributes.",
    "Error: empty tag.",
    "Error reading end tag.",
    "Error parsing Unknown.",
    "Error parsing Comment.",
    "Error parsing Declaration.",
    "Error document empty.",
    "Error null (0) or unexpected EOF found in input stream.",
    "Error parsing CData.",
    "Error when TiXmlElement added to document, because TiXmlElement can only be at the root.",
}
```

#### 4.26.3.2 utf8ByteTable

```
const int TiXmlBase::utf8ByteTable [static]
```

**Initial value:**

```
{  
  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1,  
    2,   2,   2,   2,   2,   2,   2,   2,   2,   2,   2,   2,   2,   2,   2,  
    3,   3,   3,   3,   3,   3,   3,   3,   3,   3,   3,   3,   3,   3,   3,  
    4,   4,   4,   4,   4,   1,   1,   1,   1,   1,   1,   1,   1,   1,   1  
}
```

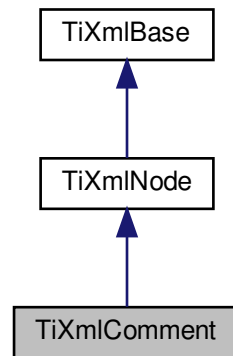
The documentation for this class was generated from the following files:

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.cpp
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxmlerror.cpp
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp

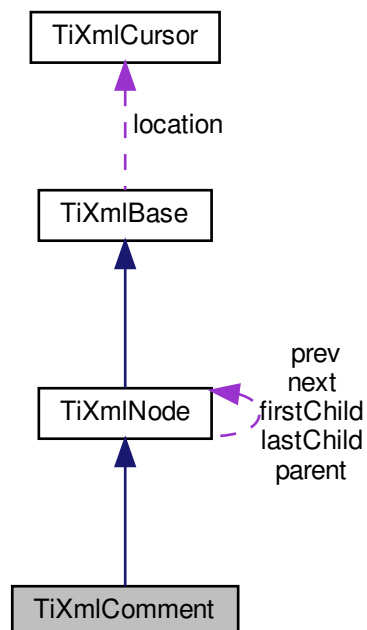
## 4.27 TiXmlComment Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlComment:



Collaboration diagram for TiXmlComment:





## Public Member Functions

- [TiXmlComment](#) ()  
*Constructs an empty comment.*
- [TiXmlComment](#) (const char \*\_value)  
*Construct a comment from text.*
- **TiXmlComment** (const [TiXmlComment](#) &)
- [TiXmlComment](#) & **operator=** (const [TiXmlComment](#) &base)
- virtual [TiXmlNode](#) \* [Clone](#) () const  
*Returns a copy of this Comment.*
- virtual void [Print](#) (FILE \*cfile, int depth) const
- virtual const char \* **Parse** (const char \*p, [TiXmlParsingData](#) \*data, [TiXmlEncoding](#) encoding)
- virtual const [TiXmlComment](#) \* [ToComment](#) () const  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual [TiXmlComment](#) \* [ToComment](#) ()  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual bool [Accept](#) ([TiXmlVisitor](#) \*visitor) const

## Protected Member Functions

- void **CopyTo** ([TiXmlComment](#) \*target) const

## Additional Inherited Members

### 4.27.1 Detailed Description

An XML comment.

### 4.27.2 Member Function Documentation

#### 4.27.2.1 [Accept\(\)](#)

```
bool TiXmlComment::Accept (
    TiXmlVisitor * visitor ) const [virtual]
```

Walk the XML tree visiting this node and all of its children.

Implements [TiXmlNode](#).

Here is the call graph for this function:



#### 4.27.2.2 Print()

```
void TiXmlComment::Print (
    FILE * cfile,
    int depth ) const [virtual]
```

All TinyXml classes can print themselves to a filestream or the string class ([TiXmlString](#) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements [TiXmlBase](#).

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

- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.cpp`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp`

## 4.28 TiXmlCursor Struct Reference

### Public Member Functions

- void **Clear** ()

### Public Attributes

- int **row**
- int **col**

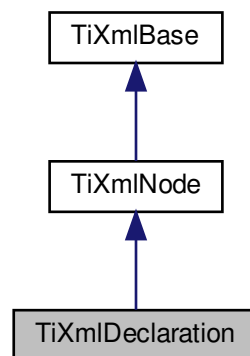
The documentation for this struct was generated from the following file:

- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`

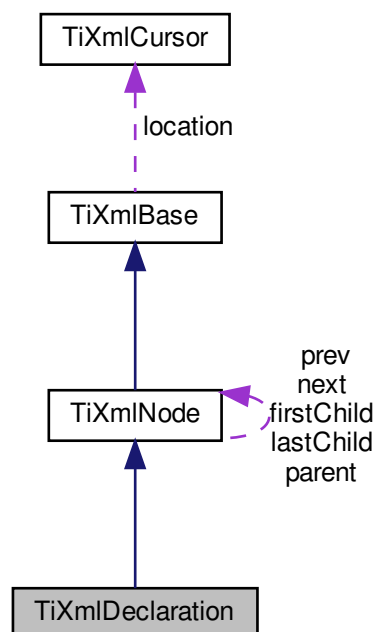
## 4.29 TiXmlDeclaration Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlDeclaration:



Collaboration diagram for TiXmlDeclaration:



## Public Member Functions

- [TiXmlDeclaration](#) ()  
*Construct an empty declaration.*
- [TiXmlDeclaration](#) (const char \*\_version, const char \*\_encoding, const char \*\_standalone)  
*Construct.*
- [TiXmlDeclaration](#) (const [TiXmlDeclaration](#) &copy)
- [TiXmlDeclaration](#) & **operator=** (const [TiXmlDeclaration](#) &copy)
- const char \* [Version](#) () const  
*Version. Will return an empty string if none was found.*
- const char \* [Encoding](#) () const  
*Encoding. Will return an empty string if none was found.*
- const char \* [Standalone](#) () const  
*Is this a standalone document?*
- virtual [TiXmlNode](#) \* [Clone](#) () const  
*Creates a copy of this Declaration and returns it.*
- virtual void **Print** (FILE \*cfile, int depth, TIXML\_STRING \*str) const
- virtual void [Print](#) (FILE \*cfile, int depth) const
- virtual const char \* **Parse** (const char \*p, [TiXmlParsingData](#) \*data, TiXmlEncoding encoding)
- virtual const [TiXmlDeclaration](#) \* [ToDeclaration](#) () const  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual [TiXmlDeclaration](#) \* [ToDeclaration](#) ()  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual bool [Accept](#) ([TiXmlVisitor](#) \*visitor) const

## Protected Member Functions

- void **CopyTo** ([TiXmlDeclaration](#) \*target) const

## Additional Inherited Members

### 4.29.1 Detailed Description

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

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

TinyXml will happily read or write files without a declaration, however. There are 3 possible attributes to the declaration: version, encoding, and standalone.

Note: In this version of the code, the attributes are handled as special cases, not generic attributes, simply because there can only be at most 3 and they are always the same.

### 4.29.2 Member Function Documentation

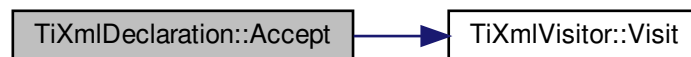
#### 4.29.2.1 Accept()

```
bool TiXmlDeclaration::Accept (
    TiXmlVisitor * visitor ) const [virtual]
```

Walk the XML tree visiting this node and all of its children.

Implements [TiXmlNode](#).

Here is the call graph for this function:



#### 4.29.2.2 Print()

```
virtual void TiXmlDeclaration::Print (
    FILE * cfile,
    int depth ) const [inline], [virtual]
```

All TinyXml classes can print themselves to a filestream or the string class ([TiXmlString](#) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements [TiXmlBase](#).

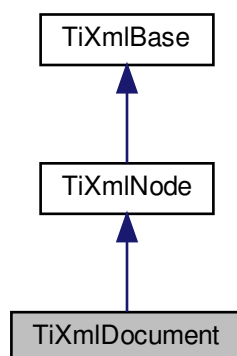
The documentation for this class was generated from the following files:

- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.cpp`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp`

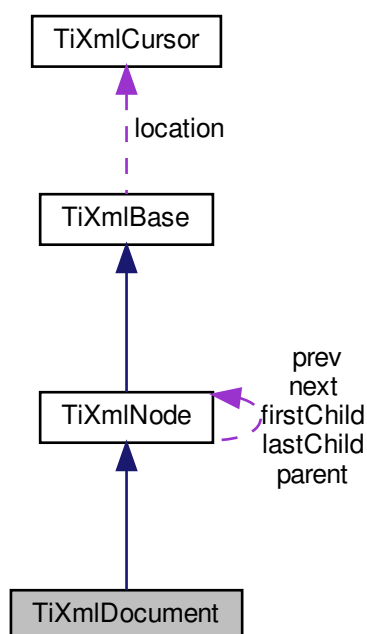
### 4.30 TiXmlDocument Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlDocument:



Collaboration diagram for TiXmlDocument:



## Public Member Functions

- [TiXmlDocument](#) ()  
*Create an empty document, that has no name.*
- [TiXmlDocument](#) (const char \*documentName)  
*Create a document with a name. The name of the document is also the filename of the xml.*
- [TiXmlDocument](#) (const [TiXmlDocument](#) &copy)
- [TiXmlDocument](#) & **operator=** (const [TiXmlDocument](#) &copy)
- bool [LoadFile](#) (TiXmlEncoding encoding=TIXML\_DEFAULT\_ENCODING)
- bool [SaveFile](#) () const  
*Save a file using the current document value. Returns true if successful.*
- bool [LoadFile](#) (const char \*filename, TiXmlEncoding encoding=TIXML\_DEFAULT\_ENCODING)  
*Load a file using the given filename. Returns true if successful.*
- bool [SaveFile](#) (const char \*filename) const  
*Save a file using the given filename. Returns true if successful.*
- bool [LoadFile](#) (FILE \*, TiXmlEncoding encoding=TIXML\_DEFAULT\_ENCODING)
- bool [SaveFile](#) (FILE \*) const  
*Save a file using the given FILE\*. Returns true if successful.*
- virtual const char \* [Parse](#) (const char \*p, [TiXmlParsingData](#) \*data=0, TiXmlEncoding encoding=TIXML\_DEFAULT\_ENCODING)
- const [TiXmlElement](#) \* [RootElement](#) () const
- [TiXmlElement](#) \* [RootElement](#) ()
- bool [Error](#) () const
- const char \* [ErrorDesc](#) () const  
*Contains a textual (english) description of the error if one occurs.*
- int [ErrorId](#) () const
- int [ErrorRow](#) () const
- int [ErrorCol](#) () const  
*The column where the error occurred. See [ErrorRow\(\)](#)*
- void [SetTabSize](#) (int \_tabsize)
- int [TabSize](#) () const
- void [ClearError](#) ()
- void [Print](#) () const
- virtual void [Print](#) (FILE \*cfile, int depth=0) const  
*Print this Document to a FILE stream.*
- void [SetError](#) (int err, const char \*errorLocation, [TiXmlParsingData](#) \*prevData, TiXmlEncoding encoding)
- virtual const [TiXmlDocument](#) \* [ToDocument](#) () const  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual [TiXmlDocument](#) \* [ToDocument](#) ()  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual bool [Accept](#) ([TiXmlVisitor](#) \*content) const

## Protected Member Functions

- virtual [TiXmlNode](#) \* [Clone](#) () const

## Additional Inherited Members

### 4.30.1 Detailed Description

Always the top level node. A document binds together all the XML pieces. It can be saved, loaded, and printed to the screen. The 'value' of a document node is the xml file name.

## 4.30.2 Member Function Documentation

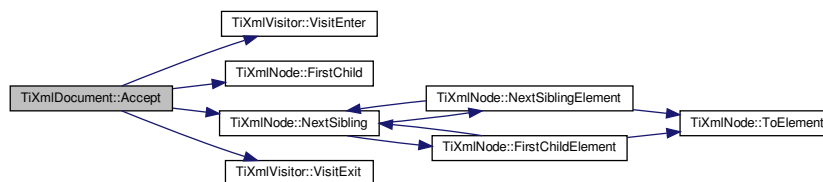
### 4.30.2.1 Accept()

```
bool TiXmlDocument::Accept (
    TiXmlVisitor * content ) const [virtual]
```

Walk the XML tree visiting this node and all of its children.

Implements [TiXmlNode](#).

Here is the call graph for this function:



### 4.30.2.2 ClearError()

```
void TiXmlDocument::ClearError ( ) [inline]
```

If you have handled the error, it can be reset with this call. The error state is automatically cleared if you Parse a new XML block.

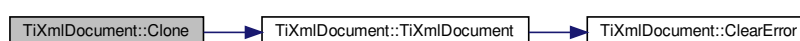
### 4.30.2.3 Clone()

```
TiXmlNode * TiXmlDocument::Clone ( ) const [protected], [virtual]
```

Create an exact duplicate of this node and return it. The memory must be deleted by the caller.

Implements [TiXmlNode](#).

Here is the call graph for this function:





#### 4.30.2.4 Error()

```
bool TiXmlDocument::Error ( ) const [inline]
```

If an error occurs, Error will be set to true. Also,

- The [ErrorId\(\)](#) will contain the integer identifier of the error (not generally useful)
- The [ErrorDesc\(\)](#) method will return the name of the error. (very useful)
- The [ErrorRow\(\)](#) and [ErrorCol\(\)](#) will return the location of the error (if known)

#### 4.30.2.5 ErrorId()

```
int TiXmlDocument::ErrorId ( ) const [inline]
```

Generally, you probably want the error string ( [ErrorDesc\(\)](#) ). But if you prefer the ErrorId, this function will fetch it.

#### 4.30.2.6 ErrorRow()

```
int TiXmlDocument::ErrorRow ( ) const [inline]
```

Returns the location (if known) of the error. The first column is column 1, and the first row is row 1. A value of 0 means the row and column wasn't applicable (memory errors, for example, have no row/column) or the parser lost the error. (An error in the error reporting, in that case.)

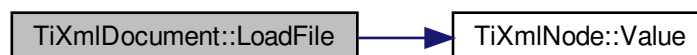
See also

[SetTabSize](#), [Row](#), [Column](#)

#### 4.30.2.7 LoadFile() [1/2]

```
bool TiXmlDocument::LoadFile (
    TiXmlEncoding encoding = TIXML_DEFAULT_ENCODING )
```

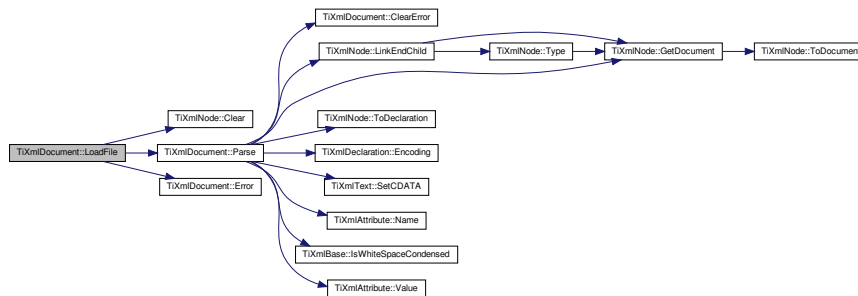
Load a file using the current document value. Returns true if successful. Will delete any existing document data before loading. Here is the call graph for this function:



#### 4.30.2.8 LoadFile() [2/2]

```
bool TiXmlDocument::LoadFile (
    FILE * file,
    TiXmlEncoding encoding = TIXML_DEFAULT_ENCODING )
```

Load a file using the given FILE\*. Returns true if successful. Note that this method doesn't stream - the entire object pointed at by the FILE\* will be interpreted as an XML file. TinyXML doesn't stream in XML from the current file location. Streaming may be added in the future. Here is the call graph for this function:



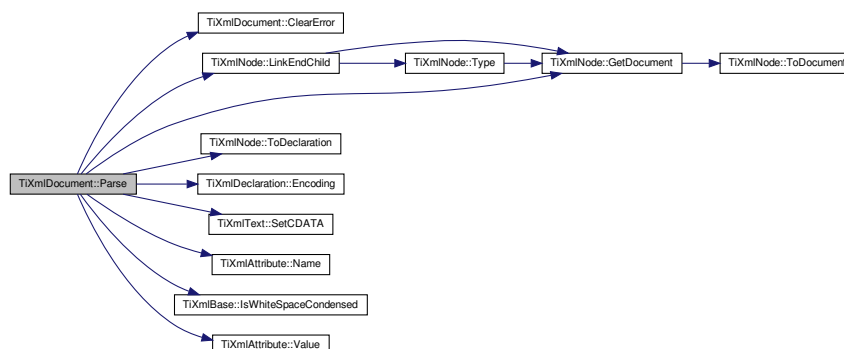
#### 4.30.2.9 Parse()

```
const char * TiXmlDocument::Parse (
    const char * p,
    TiXmlParsingData * data = 0,
    TiXmlEncoding encoding = TIXML_DEFAULT_ENCODING ) [virtual]
```

Parse the given null terminated block of xml data. Passing in an encoding to this method (either TIXML\_ENCODING\_LEGACY or TIXML\_ENCODING\_UTF8 will force TinyXml to use that encoding, regardless of what TinyXml might otherwise try to detect.

Implements [TiXmlBase](#).

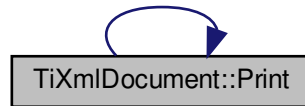
Here is the call graph for this function:



## 4.30.2.10 Print()

```
void TiXmlDocument::Print ( ) const [inline]
```

Write the document to standard out using formatted printing ("pretty print"). Here is the call graph for this function:



## 4.30.2.11 RootElement()

```
const TiXmlElement* TiXmlDocument::RootElement ( ) const [inline]
```

Get the root element – the only top level element – of the document. In well formed XML, there should only be one. TinyXml is tolerant of multiple elements at the document level.

## 4.30.2.12 SetTabSize()

```
void TiXmlDocument::SetTabSize (
    int _tabsize ) [inline]
```

[SetTabSize\(\)](#) allows the error reporting functions ([ErrorRow\(\)](#) and [ErrorCol\(\)](#)) to report the correct values for row and column. It does not change the output or input in any way.

By calling this method, with a tab size greater than 0, the row and column of each node and attribute is stored when the file is loaded. Very useful for tracking the DOM back in to the source file.

The tab size is required for calculating the location of nodes. If not set, the default of 4 is used. The tabsize is set per document. Setting the tabsize to 0 disables row/column tracking.

Note that row and column tracking is not supported when using operator<>.

The tab size needs to be enabled before the parse or load. Correct usage:

```
TiXmlDocument doc;
doc.SetTabSize( 8 );
doc.Load( "myfile.xml" );
```

## See also

[Row](#), [Column](#)

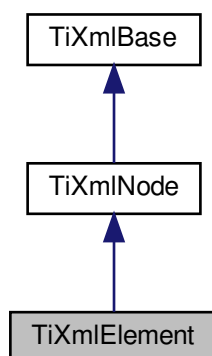
The documentation for this class was generated from the following files:

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.cpp
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp

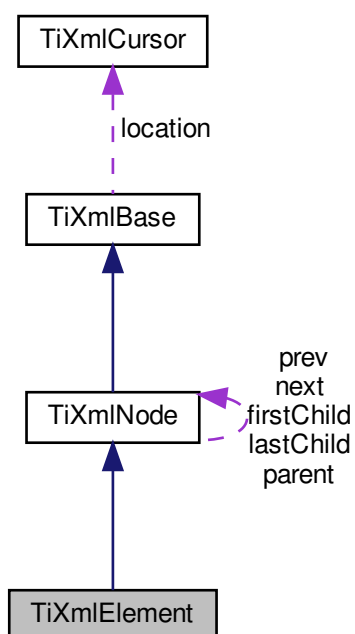
## 4.31 TiXmlElement Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlElement:



Collaboration diagram for TiXmlElement:



## Public Member Functions

- [TiXmlElement](#) (const char \*in\_value)  
*Construct an element.*
- [TiXmlElement](#) (const [TiXmlElement](#) &)
- [TiXmlElement](#) & **operator=** (const [TiXmlElement](#) &base)
- const char \* [Attribute](#) (const char \*name) const
- const char \* [Attribute](#) (const char \*name, int \*i) const
- const char \* [Attribute](#) (const char \*name, double \*d) const
- int [QueryIntAttribute](#) (const char \*name, int \*\_value) const
- int [QueryUnsignedAttribute](#) (const char \*name, unsigned \*\_value) const  
*QueryUnsignedAttribute examines the attribute - see [QueryIntAttribute\(\)](#).*
- int [QueryBoolAttribute](#) (const char \*name, bool \*\_value) const
- int [QueryDoubleAttribute](#) (const char \*name, double \*\_value) const  
*QueryDoubleAttribute examines the attribute - see [QueryIntAttribute\(\)](#).*
- int [QueryFloatAttribute](#) (const char \*name, float \*\_value) const  
*QueryFloatAttribute examines the attribute - see [QueryIntAttribute\(\)](#).*
- void [SetAttribute](#) (const char \*name, const char \*\_value)
- void [SetAttribute](#) (const char \*name, int value)
- void [SetDoubleAttribute](#) (const char \*name, double value)
- void [RemoveAttribute](#) (const char \*name)
- const [TiXmlAttribute](#) \* [FirstAttribute](#) () const  
*Access the first attribute in this element.*
- [TiXmlAttribute](#) \* **FirstAttribute** ()
- const [TiXmlAttribute](#) \* [LastAttribute](#) () const  
*Access the last attribute in this element.*
- [TiXmlAttribute](#) \* **LastAttribute** ()
- const char \* [GetText](#) () const
- virtual [TiXmlNode](#) \* [Clone](#) () const  
*Creates a new Element and returns it - the returned element is a copy.*
- virtual void [Print](#) (FILE \*cfile, int depth) const
- virtual const char \* **Parse** (const char \*p, [TiXmlParsingData](#) \*data, [TiXmlEncoding](#) encoding)
- virtual const [TiXmlElement](#) \* [ToElement](#) () const  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual [TiXmlElement](#) \* **ToElement** ()  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual bool [Accept](#) ([TiXmlVisitor](#) \*visitor) const

## Protected Member Functions

- void **CopyTo** ([TiXmlElement](#) \*target) const
- void **ClearThis** ()
- const char \* **ReadValue** (const char \*in, [TiXmlParsingData](#) \*prevData, [TiXmlEncoding](#) encoding)

## Additional Inherited Members

### 4.31.1 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.

## 4.31.2 Member Function Documentation

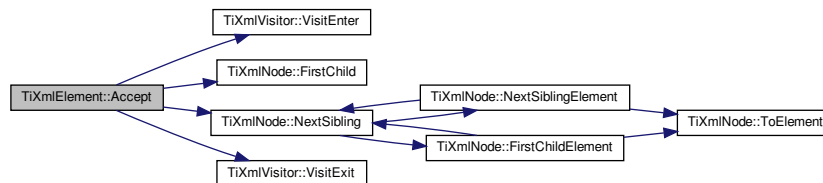
### 4.31.2.1 Accept()

```
bool TiXmlElement::Accept (
    TiXmlVisitor * visitor ) const [virtual]
```

Walk the XML tree visiting this node and all of its children.

Implements [TiXmlNode](#).

Here is the call graph for this function:



### 4.31.2.2 Attribute() [1/3]

```
const char * TiXmlElement::Attribute (
    const char * name ) const
```

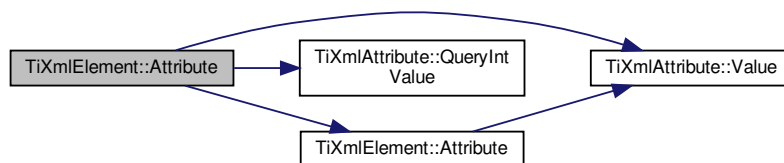
Given an attribute name, [Attribute\(\)](#) returns the value for the attribute of that name, or null if none exists. Here is the call graph for this function:



## 4.31.2.3 Attribute() [2/3]

```
const char * TiXmlElement::Attribute (
    const char * name,
    int * i ) const
```

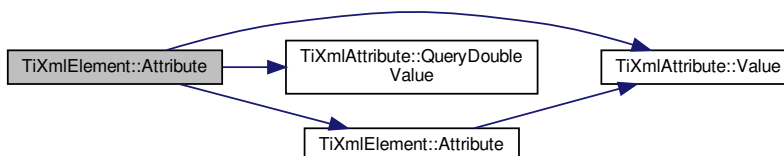
Given an attribute name, [Attribute\(\)](#) returns the value for the attribute of that name, or null if none exists. If the attribute exists and can be converted to an integer, the integer value will be put in the return 'i', if 'i' is non-null. Here is the call graph for this function:



## 4.31.2.4 Attribute() [3/3]

```
const char * TiXmlElement::Attribute (
    const char * name,
    double * d ) const
```

Given an attribute name, [Attribute\(\)](#) returns the value for the attribute of that name, or null if none exists. If the attribute exists and can be converted to a double, the double value will be put in the return 'd', if 'd' is non-null. Here is the call graph for this function:



#### 4.31.2.5 GetText()

```
const char * TiXmlElement::GetText ( ) const
```

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

If the first child of 'this' is a [TiXmlText](#), 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:

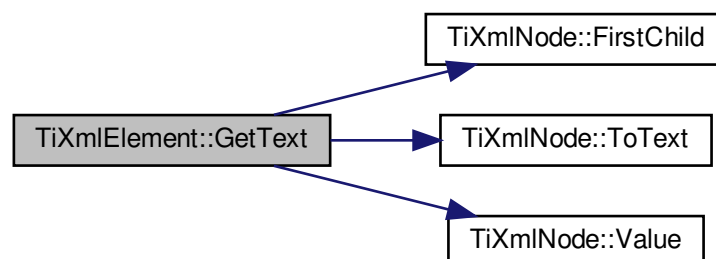
```
<foo><b>This is text</b></foo>
```

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 ".

WARNING: [GetText\(\)](#) accesses a child node - don't become confused with the similarly named [TiXmlHandle::Text\(\)](#) and [TiXmlNode::ToText\(\)](#) which are safe type casts on the referenced node. Here is the call graph for this function:





## 4.31.2.6 Print()

```
void TiXmlElement::Print (
    FILE * cfile,
    int depth ) const [virtual]
```

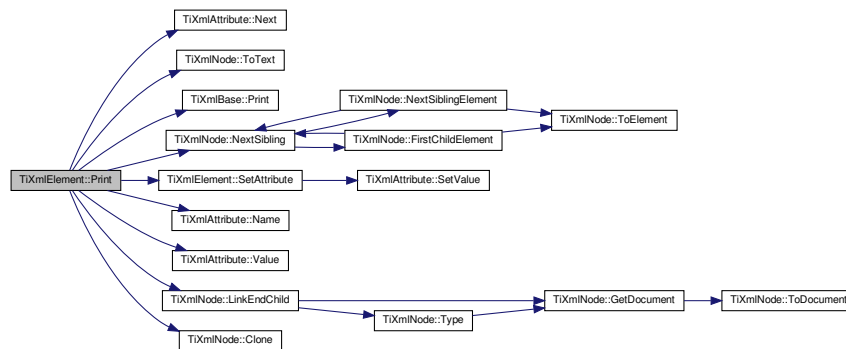
All TinyXml classes can print themselves to a filestream or the string class ([TiXmlString](#) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements [TiXmlBase](#).

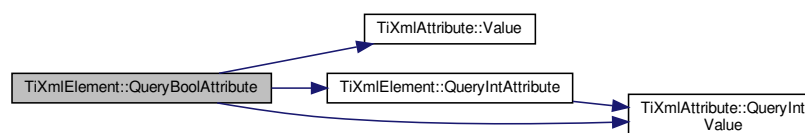
Here is the call graph for this function:



## 4.31.2.7 QueryBoolAttribute()

```
int TiXmlElement::QueryBoolAttribute (
    const char * name,
    bool * _value ) const
```

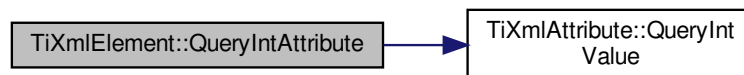
`QueryBoolAttribute` examines the attribute - see [QueryIntAttribute\(\)](#). Note that '1', 'true', or 'yes' are considered true, while '0', 'false' and 'no' are considered false. Here is the call graph for this function:



#### 4.31.2.8 QueryIntAttribute()

```
int TiXmlElement::QueryIntAttribute (
    const char * name,
    int * _value ) const
```

QueryIntAttribute examines the attribute - it is an alternative to the [Attribute\(\)](#) method with richer error checking. If the attribute is an integer, it is stored in 'value' and the call returns TIXML\_SUCCESS. If it is not an integer, it returns TIXML\_WRONG\_TYPE. If the attribute does not exist, then TIXML\_NO\_ATTRIBUTE is returned. Here is the call graph for this function:



#### 4.31.2.9 RemoveAttribute()

```
void TiXmlElement::RemoveAttribute (
    const char * name )
```

Deletes an attribute with the given name.

#### 4.31.2.10 SetAttribute() [1/2]

```
void TiXmlElement::SetAttribute (
    const char * name,
    const char * _value )
```

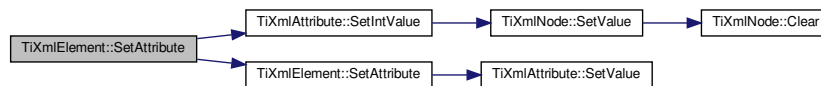
Sets an attribute of name to a given value. The attribute will be created if it does not exist, or changed if it does. Here is the call graph for this function:



## 4.31.2.11 SetAttribute() [2/2]

```
void TiXmlElement::SetAttribute (
    const char * name,
    int value )
```

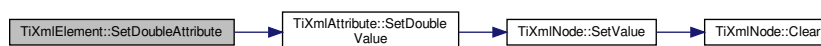
Sets an attribute of name to a given value. The attribute will be created if it does not exist, or changed if it does. Here is the call graph for this function:



## 4.31.2.12 SetDoubleAttribute()

```
void TiXmlElement::SetDoubleAttribute (
    const char * name,
    double value )
```

Sets an attribute of name to a given value. The attribute will be created if it does not exist, or changed if it does. Here is the call graph for this function:



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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.cpp
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp

## 4.32 TiXmlHandle Class Reference

```
#include <tinyxml.h>
```

## Public Member Functions

- [TiXmlHandle](#) ([TiXmlNode](#) \* \_node)  
*Create a handle from any node (at any depth of the tree.) This can be a null pointer.*
- [TiXmlHandle](#) (const [TiXmlHandle](#) &ref)  
*Copy constructor.*
- [TiXmlHandle](#) **operator=** (const [TiXmlHandle](#) &ref)
- [TiXmlHandle](#) [FirstChild](#) () const  
*Return a handle to the first child node.*
- [TiXmlHandle](#) [FirstChild](#) (const char \*value) const  
*Return a handle to the first child node with the given name.*
- [TiXmlHandle](#) [FirstChildElement](#) () const  
*Return a handle to the first child element.*
- [TiXmlHandle](#) [FirstChildElement](#) (const char \*value) const  
*Return a handle to the first child element with the given name.*
- [TiXmlHandle](#) [Child](#) (const char \*value, int index) const
- [TiXmlHandle](#) [Child](#) (int index) const
- [TiXmlHandle](#) [ChildElement](#) (const char \*value, int index) const
- [TiXmlHandle](#) [ChildElement](#) (int index) const
- [TiXmlNode](#) \* [ToNode](#) () const
- [TiXmlElement](#) \* [ToElement](#) () const
- [TiXmlText](#) \* [ToText](#) () const
- [TiXmlUnknown](#) \* [ToUnknown](#) () const
- [TiXmlNode](#) \* [Node](#) () const
- [TiXmlElement](#) \* [Element](#) () const
- [TiXmlText](#) \* [Text](#) () const
- [TiXmlUnknown](#) \* [Unknown](#) () const

### 4.32.1 Detailed Description

A [TiXmlHandle](#) is a class that wraps a node pointer with null checks; this is an incredibly useful thing. Note that [TiXmlHandle](#) is not part of the TinyXml 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:

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

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

```
TiXmlHandle docHandle( &document );
TiXmlElement* child2 = docHandle.FirstChild( "Document" ).FirstChild( "Element" ).Child( "Child", 1 ).ToElement();
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.

```
TiXmlHandle handleCopy = handle;
```

What they should not be used for is iteration:

```
int i=0;
while ( true )
{
    TiXmlElement* child = docHandle.FirstChild( "Document" ).FirstChild( "Element" ).Child( "Child", i ).ToElement();
    if ( !child )
        break;
    // do something
    ++i;
}
```

It seems reasonable, but it is in fact two embedded while loops. The Child method is a linear walk to find the element, so this code would iterate much more than it needs to. Instead, prefer:

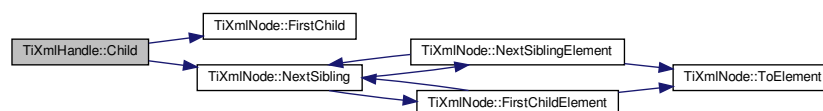
```
TiXmlElement* child = docHandle.FirstChild( "Document" ).FirstChild( "Element" ).FirstChild( "Child" ).ToElement();
for( child; child; child=child->NextSiblingElement() )
{
    // do something
}
```

## 4.32.2 Member Function Documentation

### 4.32.2.1 Child() [1/2]

```
TiXmlHandle TiXmlHandle::Child (
    const char * value,
    int index ) const
```

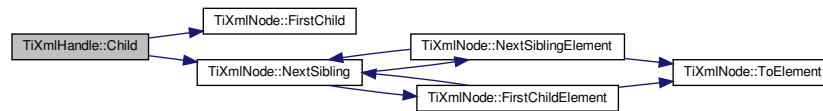
Return a handle to the "index" child with the given name. The first child is 0, the second 1, etc. Here is the call graph for this function:



## 4.32.2.2 Child() [2/2]

```
TiXmlHandle TiXmlHandle::Child (
    int index ) const
```

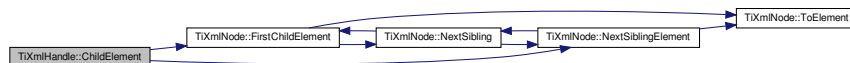
Return a handle to the "index" child. The first child is 0, the second 1, etc. Here is the call graph for this function:



## 4.32.2.3 ChildElement() [1/2]

```
TiXmlHandle TiXmlHandle::ChildElement (
    const char * value,
    int index ) const
```

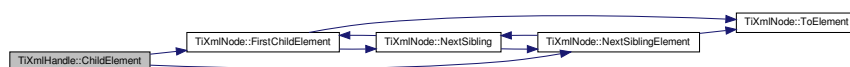
Return a handle to the "index" child element with the given name. The first child element is 0, the second 1, etc. Note that only TiXmlElement's are indexed: other types are not counted. Here is the call graph for this function:



## 4.32.2.4 ChildElement() [2/2]

```
TiXmlHandle TiXmlHandle::ChildElement (
    int index ) const
```

Return a handle to the "index" child element. The first child element is 0, the second 1, etc. Note that only TiXmlElement's are indexed: other types are not counted. Here is the call graph for this function:



#### 4.32.2.5 Element()

```
TiXmlElement* TiXmlHandle::Element ( ) const [inline]
```

**Deprecated** use ToElement. Return the handle as a [TiXmlElement](#). This may return null.

#### 4.32.2.6 Node()

```
TiXmlNode* TiXmlHandle::Node ( ) const [inline]
```

**Deprecated** use ToNode. Return the handle as a [TiXmlNode](#). This may return null.

#### 4.32.2.7 Text()

```
TiXmlText* TiXmlHandle::Text ( ) const [inline]
```

**Deprecated** use ToText(). Return the handle as a [TiXmlText](#). This may return null.

#### 4.32.2.8 ToElement()

```
TiXmlElement* TiXmlHandle::ToElement ( ) const [inline]
```

Return the handle as a [TiXmlElement](#). This may return null.

#### 4.32.2.9 ToNode()

```
TiXmlNode* TiXmlHandle::ToNode ( ) const [inline]
```

Return the handle as a [TiXmlNode](#). This may return null.

#### 4.32.2.10 ToText()

```
TiXmlText* TiXmlHandle::ToText ( ) const [inline]
```

Return the handle as a [TiXmlText](#). This may return null.

## 4.32.2.11 ToUnknown()

```
TiXmlUnknown* TiXmlHandle::ToUnknown ( ) const [inline]
```

Return the handle as a [TiXmlUnknown](#). This may return null.

## 4.32.2.12 Unknown()

```
TiXmlUnknown* TiXmlHandle::Unknown ( ) const [inline]
```

**Deprecated** use [ToUnknown\(\)](#) Return the handle as a [TiXmlUnknown](#). This may return null.

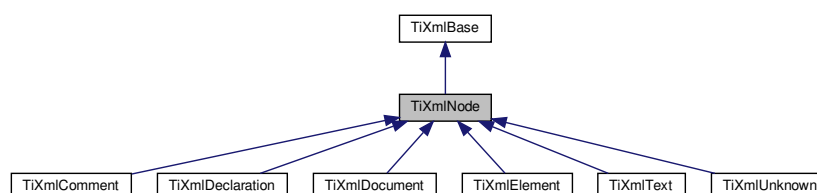
The documentation for this class was generated from the following files:

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.cpp

## 4.33 TiXmlNode Class Reference

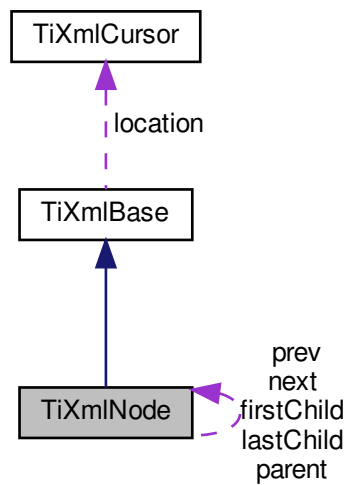
```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlNode:





Collaboration diagram for TiXmlNode:



## Public Types

- enum [NodeType](#) {  
 TINYXML\_DOCUMENT, TINYXML\_ELEMENT, TINYXML\_COMMENT, TINYXML\_UNKNOWN,  
 TINYXML\_TEXT, TINYXML\_DECLARATION, TINYXML\_TYPECOUNT }

## Public Member Functions

- const char \* [Value](#) () const
- const TIXML\_STRING & [ValueTStr](#) () const
- void [SetValue](#) (const char \* \_value)
- void [Clear](#) ()  
*Delete all the children of this node. Does not affect 'this'.*
- [TiXmlNode](#) \* [Parent](#) ()  
*One step up the DOM.*
- const [TiXmlNode](#) \* [Parent](#) () const
- const [TiXmlNode](#) \* [FirstChild](#) () const  
*The first child of this node. Will be null if there are no children.*
- [TiXmlNode](#) \* [FirstChild](#) ()
- const [TiXmlNode](#) \* [FirstChild](#) (const char \*value) const
- [TiXmlNode](#) \* [FirstChild](#) (const char \* \_value)  
*The first child of this node with the matching 'value'. Will be null if none found.*
- const [TiXmlNode](#) \* [LastChild](#) () const
- [TiXmlNode](#) \* [LastChild](#) ()  
*The last child of this node. Will be null if there are no children.*
- const [TiXmlNode](#) \* [LastChild](#) (const char \*value) const
- [TiXmlNode](#) \* [LastChild](#) (const char \* \_value)  
*The last child of this node matching 'value'. Will be null if there are no children.*

- const [TiXmlNode](#) \* [IterateChildren](#) (const [TiXmlNode](#) \*previous) const
- [TiXmlNode](#) \* [IterateChildren](#) (const [TiXmlNode](#) \*previous)
- const [TiXmlNode](#) \* [IterateChildren](#) (const char \*value, const [TiXmlNode](#) \*previous) const  
*This flavor of IterateChildren searches for children with a particular 'value'.*
- [TiXmlNode](#) \* [IterateChildren](#) (const char \*\_value, const [TiXmlNode](#) \*previous)
- [TiXmlNode](#) \* [InsertEndChild](#) (const [TiXmlNode](#) &addThis)
- [TiXmlNode](#) \* [LinkEndChild](#) ([TiXmlNode](#) \*addThis)
- [TiXmlNode](#) \* [InsertBeforeChild](#) ([TiXmlNode](#) \*beforeThis, const [TiXmlNode](#) &addThis)
- [TiXmlNode](#) \* [InsertAfterChild](#) ([TiXmlNode](#) \*afterThis, const [TiXmlNode](#) &addThis)
- [TiXmlNode](#) \* [ReplaceChild](#) ([TiXmlNode](#) \*replaceThis, const [TiXmlNode](#) &withThis)
- bool [RemoveChild](#) ([TiXmlNode](#) \*removeThis)  
*Delete a child of this node.*
- const [TiXmlNode](#) \* [PreviousSibling](#) () const  
*Navigate to a sibling node.*
- [TiXmlNode](#) \* [PreviousSibling](#) ()
- const [TiXmlNode](#) \* [PreviousSibling](#) (const char \*) const  
*Navigate to a sibling node.*
- [TiXmlNode](#) \* [PreviousSibling](#) (const char \*\_prev)
- const [TiXmlNode](#) \* [NextSibling](#) () const  
*Navigate to a sibling node.*
- [TiXmlNode](#) \* [NextSibling](#) ()
- const [TiXmlNode](#) \* [NextSibling](#) (const char \*) const  
*Navigate to a sibling node with the given 'value'.*
- [TiXmlNode](#) \* [NextSibling](#) (const char \*\_next)
- const [TiXmlElement](#) \* [NextSiblingElement](#) () const
- [TiXmlElement](#) \* [NextSiblingElement](#) ()
- const [TiXmlElement](#) \* [NextSiblingElement](#) (const char \*) const
- [TiXmlElement](#) \* [NextSiblingElement](#) (const char \*\_next)
- const [TiXmlElement](#) \* [FirstChildElement](#) () const  
*Convenience function to get through elements.*
- [TiXmlElement](#) \* [FirstChildElement](#) ()
- const [TiXmlElement](#) \* [FirstChildElement](#) (const char \*\_value) const  
*Convenience function to get through elements.*
- [TiXmlElement](#) \* [FirstChildElement](#) (const char \*\_value)
- int [Type](#) () const
- const [TiXmlDocument](#) \* [GetDocument](#) () const
- [TiXmlDocument](#) \* [GetDocument](#) ()
- bool [NoChildren](#) () const  
*Returns true if this node has no children.*
- virtual const [TiXmlDocument](#) \* [ToDocument](#) () const  
*Cast to a more defined type. Will return null if not of the requested type.*
- virtual const [TiXmlElement](#) \* [ToElement](#) () const  
*Cast to a more defined type. Will return null if not of the requested type.*
- virtual const [TiXmlComment](#) \* [ToComment](#) () const  
*Cast to a more defined type. Will return null if not of the requested type.*
- virtual const [TiXmlUnknown](#) \* [ToUnknown](#) () const  
*Cast to a more defined type. Will return null if not of the requested type.*
- virtual const [TiXmlText](#) \* [ToText](#) () const  
*Cast to a more defined type. Will return null if not of the requested type.*
- virtual const [TiXmlDeclaration](#) \* [ToDeclaration](#) () const  
*Cast to a more defined type. Will return null if not of the requested type.*
- virtual [TiXmlDocument](#) \* [ToDocument](#) ()

- Cast to a more defined type. Will return null if not of the requested type.*
  - virtual [TiXmlElement](#) \* [ToElement](#) ()
- Cast to a more defined type. Will return null if not of the requested type.*
  - virtual [TiXmlComment](#) \* [ToComment](#) ()
- Cast to a more defined type. Will return null if not of the requested type.*
  - virtual [TiXmlUnknown](#) \* [ToUnknown](#) ()
- Cast to a more defined type. Will return null if not of the requested type.*
  - virtual [TiXmlText](#) \* [ToText](#) ()
- Cast to a more defined type. Will return null if not of the requested type.*
  - virtual [TiXmlDeclaration](#) \* [ToDeclaration](#) ()
- Cast to a more defined type. Will return null if not of the requested type.*
  - virtual [TiXmlNode](#) \* [Clone](#) () const =0
- virtual bool [Accept](#) ([TiXmlVisitor](#) \*visitor) const =0

### Protected Member Functions

- [TiXmlNode](#) ([NodeType](#) \_type)
- void [CopyTo](#) ([TiXmlNode](#) \*target) const
- [TiXmlNode](#) \* [Identify](#) (const char \*start, [TiXmlEncoding](#) encoding)

### Protected Attributes

- [TiXmlNode](#) \* [parent](#)
- [NodeType](#) [type](#)
- [TiXmlNode](#) \* [firstChild](#)
- [TiXmlNode](#) \* [lastChild](#)
- [TIXML\\_STRING](#) [value](#)
- [TiXmlNode](#) \* [prev](#)
- [TiXmlNode](#) \* [next](#)

### Friends

- class [TiXmlDocument](#)
- class [TiXmlElement](#)

### Additional Inherited Members

#### 4.33.1 Detailed Description

The parent class for everything in the Document Object Model. (Except for attributes). Nodes have siblings, a parent, and children. A node can be in a document, or stand on its own. The type of a [TiXmlNode](#) can be queried, and it can be cast to its more defined type.

#### 4.33.2 Member Enumeration Documentation

#### 4.33.2.1 NodeType

```
enum TiXmlNode::NodeType
```

The types of XML nodes supported by TinyXml. (All the unsupported types are picked up by UNKNOWN.)

### 4.33.3 Member Function Documentation

#### 4.33.3.1 Accept()

```
virtual bool TiXmlNode::Accept (
    TiXmlVisitor * visitor ) const [pure virtual]
```

Accept a hierarchical visit the nodes in the TinyXML DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the [TiXmlVisitor](#) interface.

This is essentially a SAX interface for TinyXML. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML 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\(\)](#):

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

Implemented in [TiXmlDocument](#), [TiXmlUnknown](#), [TiXmlDeclaration](#), [TiXmlText](#), [TiXmlComment](#), and [TiXmlElement](#).

#### 4.33.3.2 Clone()

```
virtual TiXmlNode* TiXmlNode::Clone ( ) const [pure virtual]
```

Create an exact duplicate of this node and return it. The memory must be deleted by the caller.

Implemented in [TiXmlDocument](#), [TiXmlUnknown](#), [TiXmlDeclaration](#), [TiXmlText](#), [TiXmlComment](#), and [TiXmlElement](#).

## 4.33.3.3 FirstChild()

```
const TiXmlNode * TiXmlNode::FirstChild (
    const char * value ) const
```

The first child of this node with the matching 'value'. Will be null if none found. Here is the call graph for this function:



## 4.33.3.4 GetDocument()

```
const TiXmlDocument * TiXmlNode::GetDocument ( ) const
```

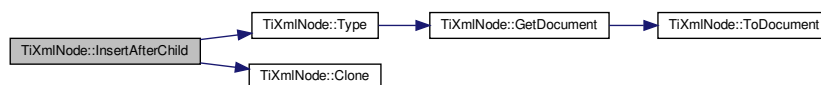
Return a pointer to the Document this node lives in. Returns null if not in a document. Here is the call graph for this function:



## 4.33.3.5 InsertAfterChild()

```
TiXmlNode * TiXmlNode::InsertAfterChild (
    TiXmlNode * afterThis,
    const TiXmlNode & addThis )
```

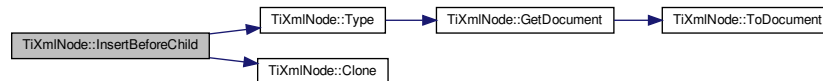
Add a new node related to this. Adds a child after the specified child. Returns a pointer to the new object or NULL if an error occurred. Here is the call graph for this function:



#### 4.33.3.6 InsertBeforeChild()

```
TiXmlNode * TiXmlNode::InsertBeforeChild (
    TiXmlNode * beforeThis,
    const TiXmlNode & addThis )
```

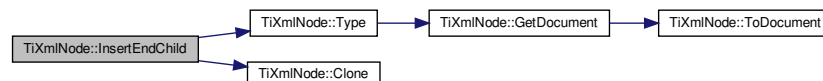
Add a new node related to this. Adds a child before the specified child. Returns a pointer to the new object or NULL if an error occurred. Here is the call graph for this function:



#### 4.33.3.7 InsertEndChild()

```
TiXmlNode * TiXmlNode::InsertEndChild (
    const TiXmlNode & addThis )
```

Add a new node related to this. Adds a child past the LastChild. Returns a pointer to the new object or NULL if an error occurred. Here is the call graph for this function:



#### 4.33.3.8 IterateChildren()

```
const TiXmlNode * TiXmlNode::IterateChildren (
    const TiXmlNode * previous ) const
```

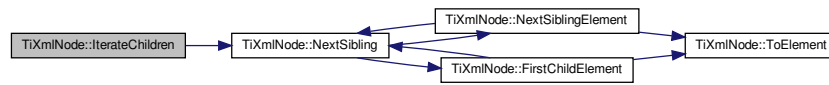
An alternate way to walk the children of a node. One way to iterate over nodes is:

```
for( child = parent->FirstChild(); child; child = child->NextSibling() )
```

`IterateChildren` does the same thing with the syntax:

```
child = 0;
while( child = parent->IterateChildren( child ) )
```

IterateChildren takes the previous child as input and finds the next one. If the previous child is null, it returns the first. IterateChildren will return null when done. Here is the call graph for this function:



#### 4.33.3.9 LinkEndChild()

```

TiXmlNode * TiXmlNode::LinkEndChild (
    TiXmlNode * addThis )

```

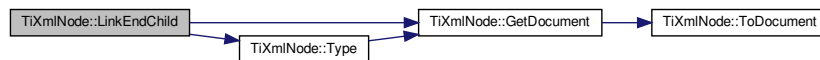
Add a new node related to this. Adds a child past the LastChild.

NOTE: the node to be added is passed by pointer, and will be henceforth owned (and deleted) by tinyXml. This method is efficient and avoids an extra copy, but should be used with care as it uses a different memory model than the other insert functions.

See also

[InsertEndChild](#)

Here is the call graph for this function:



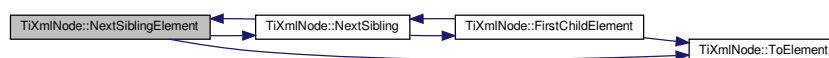
#### 4.33.3.10 NextSiblingElement() [1/2]

```

const TiXmlElement * TiXmlNode::NextSiblingElement ( ) const

```

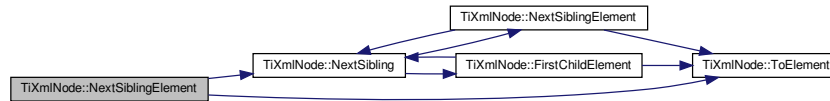
Convenience function to get through elements. Calls NextSibling and ToElement. Will skip all non-Element nodes. Returns 0 if there is not another element. Here is the call graph for this function:



#### 4.33.3.11 NextSiblingElement() [2/2]

```
const TiXmlElement * TiXmlNode::NextSiblingElement (
    const char * _value ) const
```

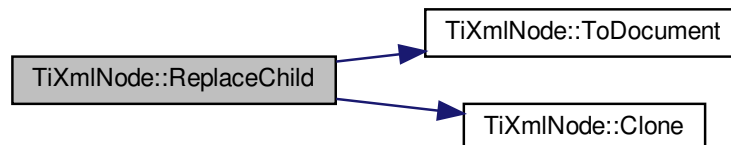
Convenience function to get through elements. Calls NextSibling and ToElement. Will skip all non-Element nodes. Returns 0 if there is not another element. Here is the call graph for this function:



#### 4.33.3.12 ReplaceChild()

```
TiXmlNode * TiXmlNode::ReplaceChild (
    TiXmlNode * replaceThis,
    const TiXmlNode & withThis )
```

Replace a child of this node. Returns a pointer to the new object or NULL if an error occurred. Here is the call graph for this function:



#### 4.33.3.13 SetValue()

```
void TiXmlNode::SetValue (
    const char * _value ) [inline]
```

Changes the value of the node. Defined as:

```
Document:  filename of the xml file
Element:   name of the element
Comment:   the comment text
Unknown:   the tag contents
Text:      the text string
```



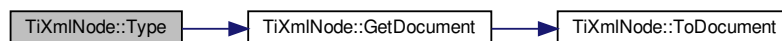
Here is the call graph for this function:



#### 4.33.3.14 Type()

```
int TiXmlNode::Type ( ) const [inline]
```

Query the type (as an enumerated value, above) of this node. The possible types are: `TINYXML_DOCUMENT`, `TINYXML_ELEMENT`, `TINYXML_COMMENT`, `TINYXML_UNKNOWN`, `TINYXML_TEXT`, and `TINYXML_DECLARATION`. Here is the call graph for this function:



#### 4.33.3.15 Value()

```
const char* TiXmlNode::Value ( ) const [inline]
```

The meaning of 'value' changes for the specific type of [TiXmlNode](#).

```
Document:  filename of the xml file
Element:   name of the element
Comment:   the comment text
Unknown:   the tag contents
Text:      the text string
```

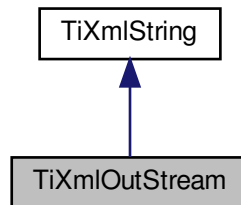
The subclasses will wrap this function.

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

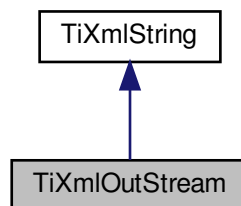
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.cpp`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp`

## 4.34 TiXmlOutStream Class Reference

Inheritance diagram for TiXmlOutStream:



Collaboration diagram for TiXmlOutStream:



### Public Member Functions

- [TiXmlOutStream](#) & **operator**<< (const [TiXmlString](#) &in)
- [TiXmlOutStream](#) & **operator**<< (const char \*in)

### Additional Inherited Members

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinystr.h

## 4.35 TiXmlParsingData Class Reference

### Public Member Functions

- void **Stamp** (const char \*now, TiXmlEncoding encoding)
- const [TiXmlCursor](#) & **Cursor** () const

## Friends

- class **TiXmlDocument**

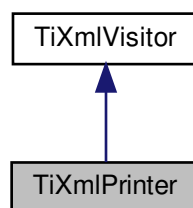
The documentation for this class was generated from the following file:

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp

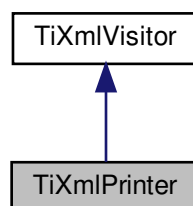
## 4.36 TiXmlPrinter Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlPrinter:



Collaboration diagram for TiXmlPrinter:



## Public Member Functions

- virtual bool [VisitEnter](#) (const [TiXmlDocument](#) &doc)  
*Visit a document.*
- virtual bool [VisitExit](#) (const [TiXmlDocument](#) &doc)  
*Visit a document.*
- virtual bool [VisitEnter](#) (const [TiXmlElement](#) &element, const [TiXmlAttribute](#) \*firstAttribute)  
*Visit an element.*
- virtual bool [VisitExit](#) (const [TiXmlElement](#) &element)  
*Visit an element.*
- virtual bool [Visit](#) (const [TiXmlDeclaration](#) &declaration)  
*Visit a declaration.*
- virtual bool [Visit](#) (const [TiXmlText](#) &text)  
*Visit a text node.*
- virtual bool [Visit](#) (const [TiXmlComment](#) &comment)  
*Visit a comment node.*
- virtual bool [Visit](#) (const [TiXmlUnknown](#) &unknown)  
*Visit an unknown node.*
- void [SetIndent](#) (const char \*\_indent)
- const char \* [Indent](#) ()  
*Query the indentation string.*
- void [SetLineBreak](#) (const char \*\_lineBreak)
- const char \* [LineBreak](#) ()  
*Query the current line breaking string.*
- void [SetStreamPrinting](#) ()
- const char \* [CStr](#) ()  
*Return the result.*
- size\_t [Size](#) ()  
*Return the length of the result string.*

### 4.36.1 Detailed Description

Print to memory functionality. The [TiXmlPrinter](#) is useful when you need to:

1. Print to memory (especially in non-STL mode)
2. Control formatting (line endings, etc.)

When constructed, the [TiXmlPrinter](#) is in its default "pretty printing" mode. Before calling [Accept\(\)](#) you can call methods to control the printing of the XML document. After [TiXmlNode::Accept\(\)](#) is called, the printed document can be accessed via the [CStr\(\)](#), [Str\(\)](#), and [Size\(\)](#) methods.

[TiXmlPrinter](#) uses the Visitor API.

```
TiXmlPrinter printer;
printer.SetIndent( "\t" );

doc.Accept( &printer );
fprintf( stdout, "%s", printer.CStr() );
```

## 4.36.2 Member Function Documentation

### 4.36.2.1 SetIndent()

```
void TiXmlPrinter::SetIndent (
    const char * _indent ) [inline]
```

Set the indent characters for printing. By default 4 spaces but tab () is also useful, or null/empty string for no indentation.

### 4.36.2.2 SetLineBreak()

```
void TiXmlPrinter::SetLineBreak (
    const char * _lineBreak ) [inline]
```

Set the line breaking string. By default set to newline ( ). Some operating systems prefer other characters, or can be set to the null/empty string for no indenation.

### 4.36.2.3 SetStreamPrinting()

```
void TiXmlPrinter::SetStreamPrinting ( ) [inline]
```

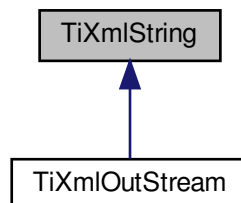
Switch over to "stream printing" which is the most dense formatting without linebreaks. Common when the XML is needed for network transmission.

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.cpp

## 4.37 TiXmlString Class Reference

Inheritance diagram for TiXmlString:



## Public Types

- typedef size\_t **size\_type**

## Public Member Functions

- **TiXmlString** (const [TiXmlString](#) &copy)
- TIXML\_EXPLICIT **TiXmlString** (const char \*copy)
- TIXML\_EXPLICIT **TiXmlString** (const char \*str, size\_type len)
- [TiXmlString](#) & **operator=** (const char \*copy)
- [TiXmlString](#) & **operator=** (const [TiXmlString](#) &copy)
- [TiXmlString](#) & **operator+=** (const char \*suffix)
- [TiXmlString](#) & **operator+=** (char single)
- [TiXmlString](#) & **operator+=** (const [TiXmlString](#) &suffix)
- const char \* **c\_str** () const
- const char \* **data** () const
- size\_type **length** () const
- size\_type **size** () const
- bool **empty** () const
- size\_type **capacity** () const
- const char & **at** (size\_type index) const
- char & **operator[]** (size\_type index) const
- size\_type **find** (char lookup) const
- size\_type **find** (char tofind, size\_type offset) const
- void **clear** ()
- void **reserve** (size\_type cap)
- [TiXmlString](#) & **assign** (const char \*str, size\_type len)
- [TiXmlString](#) & **append** (const char \*str, size\_type len)
- void **swap** ([TiXmlString](#) &other)

## Static Public Attributes

- static const size\_type **npos** = static\_cast< TiXmlString::size\_type >(-1)

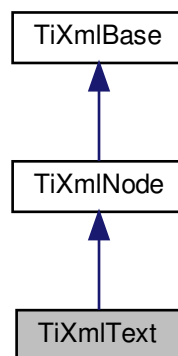
The documentation for this class was generated from the following files:

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyst.h
- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyst.cpp

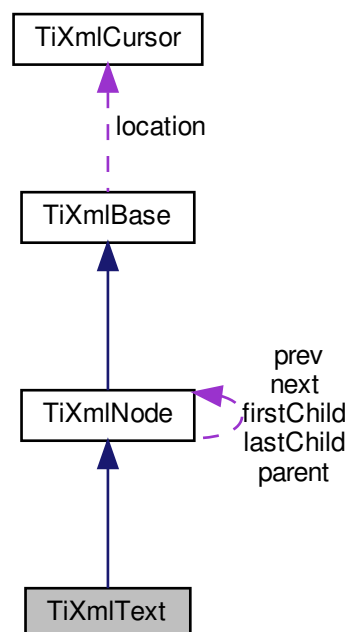
## 4.38 TiXmlText Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlText:



Collaboration diagram for TiXmlText:



## Public Member Functions

- [TiXmlText](#) (const char \*initValue)
- **TiXmlText** (const [TiXmlText](#) &copy)
- [TiXmlText](#) & **operator=** (const [TiXmlText](#) &base)
- virtual void **Print** (FILE \*cfile, int depth) const
- bool **CDATA** () const  
*Queries whether this represents text using a CDATA section.*
- void **SetCDATA** (bool \_cdata)  
*Turns on or off a CDATA representation of text.*
- virtual const char \* **Parse** (const char \*p, [TiXmlParsingData](#) \*data, TiXmlEncoding encoding)
- virtual const [TiXmlText](#) \* **ToText** () const  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual [TiXmlText](#) \* **ToText** ()  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual bool **Accept** ([TiXmlVisitor](#) \*content) const

## Protected Member Functions

- virtual [TiXmlNode](#) \* **Clone** () const  
*[internal use] Creates a new Element and returns it.*
- void **CopyTo** ([TiXmlText](#) \*target) const
- bool **Blank** () const

## Friends

- class **TiXmlElement**

## Additional Inherited Members

### 4.38.1 Detailed Description

XML text. 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\(\)](#).

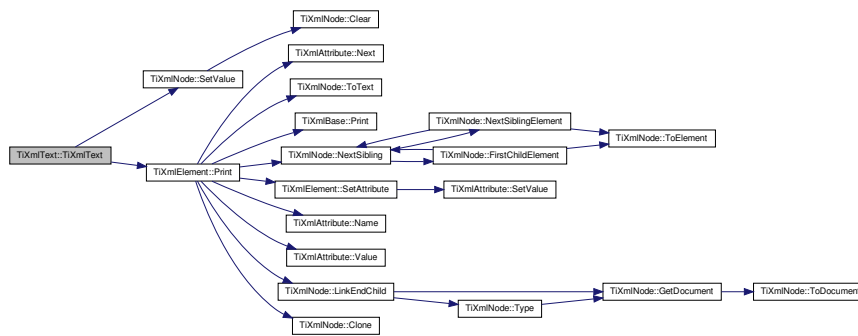
### 4.38.2 Constructor & Destructor Documentation



## 4.38.2.1 TiXmlText()

```
TiXmlText::TiXmlText (
    const char * initValue ) [inline]
```

Constructor for text element. By default, it is treated as normal, encoded text. If you want it be output as a CDATA text element, set the parameter `_cdata` to 'true'. Here is the call graph for this function:



## 4.38.3 Member Function Documentation

## 4.38.3.1 Accept()

```
bool TiXmlText::Accept (
    TiXmlVisitor * content ) const [virtual]
```

Walk the XML tree visiting this node and all of its children.

Implements [TiXmlNode](#).

Here is the call graph for this function:



#### 4.38.3.2 Print()

```
void TiXmlText::Print (
    FILE * cfile,
    int depth ) const [virtual]
```

All TinyXml classes can print themselves to a filestream or the string class ([TiXmlString](#) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements [TiXmlBase](#).

Here is the call graph for this function:



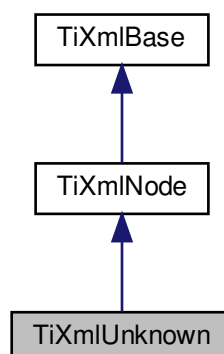
The documentation for this class was generated from the following files:

- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.cpp`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp`

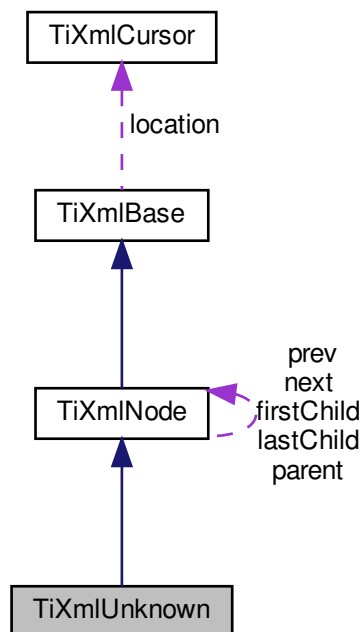
## 4.39 TiXmlUnknown Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for `TiXmlUnknown`:



Collaboration diagram for TiXmlUnknown:



## Public Member Functions

- **TiXmlUnknown** (const [TiXmlUnknown](#) &copy)
- [TiXmlUnknown](#) & **operator=** (const [TiXmlUnknown](#) &copy)
- virtual [TiXmlNode](#) \* **Clone** () const  
*Creates a copy of this Unknown and returns it.*
- virtual void **Print** (FILE \*cfile, int depth) const
- virtual const char \* **Parse** (const char \*p, [TiXmlParsingData](#) \*data, [TiXmlEncoding](#) encoding)
- virtual const [TiXmlUnknown](#) \* **ToUnknown** () const  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual [TiXmlUnknown](#) \* **ToUnknown** ()  
*Cast to a more defined type. Will return null not of the requested type.*
- virtual bool **Accept** ([TiXmlVisitor](#) \*content) const

## Protected Member Functions

- void **CopyTo** ([TiXmlUnknown](#) \*target) const

## Additional Inherited Members

### 4.39.1 Detailed Description

Any tag that tinyXml 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 TiXmlUnknowns.

## 4.39.2 Member Function Documentation

### 4.39.2.1 Accept()

```
bool TiXmlUnknown::Accept (
    TiXmlVisitor * content ) const [virtual]
```

Walk the XML tree visiting this node and all of its children.

Implements [TiXmlNode](#).

Here is the call graph for this function:



### 4.39.2.2 Print()

```
void TiXmlUnknown::Print (
    FILE * cfile,
    int depth ) const [virtual]
```

All TinyXml classes can print themselves to a filestream or the string class ([TiXmlString](#) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements [TiXmlBase](#).

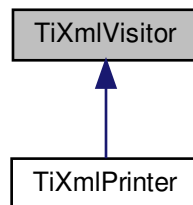
The documentation for this class was generated from the following files:

- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.h`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxml.cpp`
- `/home/jan/Promotion_linuxPC/Optim_VSTRAP/src/io/tinyXML/tinyxmlparser.cpp`

## 4.40 TiXmlVisitor Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlVisitor:



### Public Member Functions

- virtual bool [VisitEnter](#) (const [TiXmlDocument](#) &)  
*Visit a document.*
- virtual bool [VisitExit](#) (const [TiXmlDocument](#) &)  
*Visit a document.*
- virtual bool [VisitEnter](#) (const [TiXmlElement](#) &, const [TiXmlAttribute](#) \*)  
*Visit an element.*
- virtual bool [VisitExit](#) (const [TiXmlElement](#) &)  
*Visit an element.*
- virtual bool [Visit](#) (const [TiXmlDeclaration](#) &)  
*Visit a declaration.*
- virtual bool [Visit](#) (const [TiXmlText](#) &)  
*Visit a text node.*
- virtual bool [Visit](#) (const [TiXmlComment](#) &)  
*Visit a comment node.*
- virtual bool [Visit](#) (const [TiXmlUnknown](#) &)  
*Visit an unknown node.*

### 4.40.1 Detailed Description

Implements the interface to the "Visitor pattern" (see the `Accept()` method.) If you call the `Accept()` method, it requires being passed a [TiXmlVisitor](#) 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 leaves 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 [TiXmlDocument](#), although all nodes support Visiting.

You should never change the document from a callback.

See also

[TiXmlNode::Accept\(\)](#)

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

- /home/jan/Promotion\_linuxPC/Optim\_VSTRAP/src/io/tinyXML/tinyxml.h