

---

# Part I. The BoostBook Documentation Format

---

# Table of Contents

Introduction .....	4
Getting Started .....	5
Automatic setup for Unix-like systems .....	5
Manual setup for all systems .....	5
Configuring xsltproc .....	5
Configuring local DocBook XSL and DTD distributions .....	5
Configuring Doxygen for Documentation Extraction .....	6
Configuring Apache FOP .....	6
Running BoostBook .....	7
Troubleshooting .....	7
Documenting libraries .....	8
Defining a BoostBook library .....	8
From HTML to BoostBook .....	9
Sectioning in BoostBook .....	10
Bringing Together a BoostBook Document .....	11
Linking in BoostBook .....	11
Reference .....	12
BoostBook element class-specialization .....	14
BoostBook element link-test .....	15
BoostBook element link-fail-test .....	16
BoostBook element typedef .....	17
BoostBook element static-constant .....	18
BoostBook element code .....	19
BoostBook element destructor .....	20
BoostBook element template-type-parameter .....	21
BoostBook element description .....	22
BoostBook element librarylist .....	23
BoostBook element library-reference .....	24
BoostBook element boostbook .....	25
BoostBook element union .....	26
BoostBook element inherit .....	27
BoostBook element template-varargs .....	28
BoostBook element source .....	29
BoostBook element function .....	30
BoostBook element macroname .....	32
BoostBook element postconditions .....	33
BoostBook element compile-test .....	34
BoostBook element method .....	35
BoostBook element snippet .....	36
BoostBook element constructor .....	37
BoostBook element namespace .....	38
BoostBook element if-fails .....	39
BoostBook element headername .....	40
BoostBook element free-function-group .....	41
BoostBook element functionname .....	42
BoostBook element librarycategory .....	43
BoostBook element notes .....	44
BoostBook element data-member .....	45
BoostBook element specialization .....	46
BoostBook element union-specialization .....	47
BoostBook element throws .....	48
BoostBook element template-arg .....	49
BoostBook element globalname .....	50
BoostBook element method-group .....	51
BoostBook element requirement .....	52

BoostBook element precondition .....	53
BoostBook element paramtype .....	54
BoostBook element using-class .....	55
BoostBook element run-test .....	56
BoostBook element librarypurpose .....	57
BoostBook element copy-assignment .....	58
BoostBook element run-fail-test .....	59
BoostBook element template .....	60
BoostBook element compile-fail-test .....	61
BoostBook element returns .....	62
BoostBook element default .....	63
BoostBook element parameter .....	64
BoostBook element signature .....	65
BoostBook element overloaded-function .....	66
BoostBook element access .....	67
BoostBook element class .....	68
BoostBook element librarycategorydef .....	69
BoostBook element type .....	70
BoostBook element enumvalue .....	71
BoostBook element overloaded-method .....	72
BoostBook element programlisting .....	73
BoostBook element complexity .....	74
BoostBook element purpose .....	75
BoostBook element template-nontype-parameter .....	76
BoostBook element library .....	77
BoostBook element librarycategorylist .....	78
BoostBook element using-namespace .....	79
BoostBook element enumname .....	80
BoostBook element struct-specialization .....	81
BoostBook element struct .....	82
BoostBook element lib .....	83
BoostBook element enum .....	84
BoostBook element requires .....	85
BoostBook element effects .....	86
BoostBook element libraryname .....	87
BoostBook element libraryinfo .....	88
BoostBook element testsuite .....	89
BoostBook element header .....	90
BoostBook element rationale .....	91

---

# Introduction

The BoostBook documentation format is an extension of [DocBook](#), an SGML- or XML-based format for describing documentation. BoostBook augments DocBook with semantic markup that aids in the documentation of C++ libraries, specifically the [Boost C++ libraries](#), by providing the ability to express and refer to C++ constructs such as namespaces, classes, overloaded functions, templates, and specializations.

BoostBook offers additional features more specific to its use for documenting the [Boost C++ libraries](#). These features are intended to eliminate or reduce the need for duplication of information and to aid in documenting portions of Boost that might otherwise not be documented. Examples of Boost-centric features include:

- **Testsuites:** Testsuites in Boost are created by writing an appropriate Jamfile and including that Jamfile in `status/Jamfile`. If the testsuites are documented ([as in the MultiArray library](#)), the documentation is maintained separately from the testcase Jamfile, leading to duplication of information and the possibility of having the documentation out of sync with the Jamfile. BoostBook contains elements that describe a testsuite for both purposes: the BoostBook stylesheets can generate documentation for the testcases and also generate an appropriate Jamfile to integrate the testcases with the regression testing system.
- **Example programs:** Example programs in documentation need to be duplicated in testcases to ensure that the examples compile and execute correctly. Keeping the two copies in sync is a tedious and error-prone task. For instance, the following code snippet persisted for six months:

```
std::cout << f(5, 3) >> std::endl;
```

The BoostBook format allows testcases to be generated by weaving together program fragments from example programs in the documentation. This capability is integrated with testsuite generation so that example programs are normal tests in BoostBook.

---

# Getting Started

To use the Boost documentation tools, you will need several tools:

- **xsltproc**:
  - Windows with [Cygwin](#): select the `libxml2` and `libxslt` packages.
  - Windows without Cygwin: Download the [binary packages](#) from Igor Zlatkovic. At the very least, you'll need `iconv`, `zlib`, `libxml2` and `libxslt`.
  - Mac OS X with Fink: Get the `libxslt` package.
  - Mac OS X without Fink: [Download the libxslt binaries](#)
  - Any platform: [libxslt source](#).
- **doxygen**:  
Available from <http://www.doxygen.org>

## Automatic setup for Unix-like systems

BoostBook provides a nearly-automatic setup script. Once you have downloaded and installed **xsltproc**, **doxygen**, and (optionally) **java**, the setup script can download the required DocBook stylesheets, DocBook DTD, and (when Java is enabled) Apache FOP for PDF output. It will then configure Boost.Build version 2 to build BoostBook documentation.

The script requires: **sh**, **curl** and **gunzip**. To perform the installation, execute the script `tools/boostbook/setup_boostbook.sh` from a directory where you would like the resulting XSL, DTD, and Apache FOP installations to occur.

## Manual setup for all systems

This section describes how to manually configure Boost version 2 (BBv@) for BoostBook. If you can use the automatic setup script, you should. All configuration will happen in the BBv2 user configuration file, `user-config.jam`. If you do not have a copy of this file in your home directory, you should copy the one that resides in `tools/build/v2` to your home directory. Alternatively, you can edit `tools/build/v2/user-config.jam` directly or a site-wide `site-config.jam` file.

## Configuring xsltproc

To configure **xsltproc** manually, you will need to add a directive to `user-config.jam` telling it where to find **xsltproc**. If the program is in your path, just add the following line to `user-config.jam`:

```
using xsltproc ;
```

If **xsltproc** is somewhere else, use this directive, where `XSLTPROC` is the full pathname to **xsltproc** (including **xsltproc**):

```
using xsltproc : XSLTPROC ;
```

## Configuring local DocBook XSL and DTD distributions

This section describes how to configure Boost.Build to use local copies of the DocBook DTD and XSL stylesheets to improve processing time. You will first need to download two packages:

- Norman Walsh's DocBook XSL stylesheets, available at the [DocBook sourceforge site](#). Extract the DocBook XSL stylesheets to a directory on your hard disk (which we'll refer to as the `DOCBOOK_XSL_DIR`).

- The DocBook DTD, available as a ZIP archive at the [OASIS DocBook site](#). The package is called "DocBook XML 4.2". Extract the DocBook DTD to a directory on your hard disk (which we'll refer to as the `DOCBOOK_DTD_DIR`). You will want to extract this archive in a subdirectory!

Add the following directive telling BBv2 where to find the DocBook DTD and XSL stylesheets:

```
# BoostBook configuration
using boostbook
  : DOCBOOK_XSL_DIR
  : DOCBOOK_DTD_DIR
;
```

Whenever you change this directive, you will need to remove the `bin.v2` directory that BBv2 generates. This is due to longstanding bug we are trying to fix.

At this point, you should be able to build HTML documentation for libraries that do not require Doxygen. To test this, change into the directory `$BOOST_ROOT/libs/function/doc` and run the command `bjam`: it should produce HTML documentation for the Boost.Function library in the `html` subdirectory.

## Configuring Doxygen for Documentation Extraction

Doxygen is required to build the documentation for several Boost libraries. You will need a recent version of [Doxygen](#) (most of the 1.3.x and 1.4.x versions will suffice). BBv2 by adding the following directive to `user-config.jam`:

```
using doxygen : DOXYGEN ;
```

`DOXYGEN` should be replaced with the name of the **doxygen** executable (with full path name). If the right **doxygen** executable can be found via the path, this parameter can be omitted, e.g.

```
using doxygen ;
```



### Important

The relative order of declarations in `user-config.jam` / `site-config.jam` files is significant. In particular, the `using doxygen` line should come *after* the `using boostbook` declaration.

## Configuring Apache FOP

In order to generate PDF and PostScript output using Apache FOP, you will need a [Java interpreter](#) and [Apache FOP](#) (version 0.20.5 is best). Unpack Apache FOP to some directory. The top level directory of the FOP tool should contain a main script called `fop.sh` on Unix and `fop.bat` on Windows. You need to specify the location of that script and Java location to Boost.Build. Add the following to your `user-config.jam` or `site-config.jam`:

```
using fop : FOP_COMMAND
  : JAVA_HOME
;
```

replacing `FOP_COMMAND` with the full path to the FOP main script, and replacing `JAVA_HOME` with the directory where Java is installed. If the `JAVA_HOME` environment variable is already set, you don't need to specify it above.

Proper generation of images in PDFs depends on the [Jimi Image Library](#). To get FOP to use Jimi, extract the `JimiProClasses.zip` file from the Jimi SDK and rename it—if on Windows, to `jimi-1.0.jar`, or if on \*nix, to `JimiProClasses.jar`—and place it in the `lib/` subdirectory of your FOP installation.

To test PDF generation, switch to the directory `$BOOST_ROOT/libs/function/doc` and execute the command **bjam pdf**. In the absence of any errors, Apache FOP will be executed to transform the XSL:FO output of DocBook into a PDF file.

## Running BoostBook

Once BoostBook has been configured, we can build some documentation. First, change to the directory `$BOOST_ROOT/doc` and remove (or make writable) the `.html` files in `$BOOST_ROOT/doc/html`. Then, run `bjam` to build HTML documentation. You should see several warnings like these while DocBook documentation is being built from BoostBook documentation:

```
Cannot find function named 'checked_delete'
Cannot find function named 'checked_array_delete'
Cannot find function named 'next'
```

These warnings are emitted when the Boost documentation tools cannot find documentation for functions, methods, or classes that are referenced in the source, and are not harmful in any way. Once Boost.Jam has completed its execution, HTML documentation for Boost will be available in `$BOOST_ROOT/doc/html`. You can also create HTML documentation in a single (large!) HTML file with the command line `bjam onehtml`, or Unix man pages with the command line `bjam man`. The complete list of output formats is listed in [Table 1, “BoostBook Output Formats”](#). Several output formats can be passed to a single invocation of `bjam`, e.g., `bjam html man docbook` would generate HTML (multiple files), man pages, and DocBook documentation.

**Table 1. BoostBook Output Formats**

Format	Description
html	HTML output (multiple files). This is the default
onehtml	HTML output in a single HTML file.
man	Unix man pages.
pdf	PDF. Requires <a href="#">Apache FOP</a> .
ps	Postscript. Requires <a href="#">Apache FOP</a> .
docbook	<a href="#">DocBook</a> .
fo	<a href="#">XSL Formatting Objects</a>

## Troubleshooting

The Boost documentation tools are still in their early phase of development, and some things don't work as seamlessly as we would like them to, yet. In particular, error messages can be somewhat uninformative at times. If you find yourself in the situation when you have double checked everything, and yet things still don't work as expected, consider helping the tools by deleting `bin.v2` build directory.

---

# Documenting libraries

BoostBook is an extension to [DocBook](#), an XML format for representing documentation. BoostBook inherits much of its functionality and many elements from DocBook that are not redocumented here. When writing BoostBook documentation, please refer also to [DocBook: The Definitive Guide](#).

## Defining a BoostBook library

BoostBook library documentation is contained entirely within a `<library>` XML element. To create a skeletal library, we need to create a new XML document (call it `any.xml`) that contains basic information about the library. The following [BoostBook XML example](#) describes basic information about the [Boost.Any](#) library:

### Example 1. A Skeletal BoostBook Library

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE library PUBLIC "-//Boost//DTD BoostBook XML V1.0//EN"
    "http://www.boost.org/tools/boostbook/dtd/boostbook.dtd">
<library name="Any" dirname="any" xmlns:xi="http://www.w3.org/2001/XInclude"
    id="any" last-revision="$Date: 2008-07-12 20:30:45 +0100 (Sat, 12 Jul 2008) $">
  <libraryinfo>
    <author>
      <firstname>Kevlin</firstname>
      <surname>Henney</surname>
    </author>
    <librarypurpose>
      Safe, generic container for single values of different value types
    </librarypurpose>
    <librarycategory name="category:data-structures"/>
  </libraryinfo>
</library>
```

The first three lines identify this document as a BoostBook [XML](#) document. The DOCTYPE line states that the document conforms to the BoostBook DTD, and that the top-level element is a BoostBook `<library>`.

The `<library>` element actually describes the aspects of BoostBook library documentation. The attributes for the `<library>` element are:

### Attributes for the `<library>` element

name	The full name of the library, e.g., "Any"
dirname	The name of the directory, relative to <code>boost/libs</code> , in which the library resides. This name may be a relative path, such as <code>math/octonion</code> , using <code>/</code> for the directory separator.
id	A short, unique name for the library. For libraries with simple directory names (e.g., ones that do not contain a <code>/</code> ), this should be the same as the <code>dirname</code> . This <code>id</code> will be used to identify libraries and, for HTML output, will be used as the base name for the HTML file in which the library's documentation resides, so it should use only lowercase alphanumeric characters and underscores.
last-revision	Always set to <code>\$Date: 2008-07-12 20:30:45 +0100 (Sat, 12 Jul 2008) \$</code> , which is expanded by CVS to include the date and time that the file was last modified.

Inside the `<library>` element we have the `<libraryinfo>` element, which gives information about the library itself. It contains the author's name (there may be more than one `<author>` element), followed by the purpose of the library and the list of categorizations. The `<librarypurpose>` element should always contain a very short (single sentence) description of the library's purpose, and should *not* terminate with a period.



The list of categories is specified by a set of `<librarycategory>` elements. Each `<librarycategory>` element has a `name` element that identifies one of the categories. The actual list of categories is in the file `doc/src/boost.xml`.

At this point, we can apply the BoostBook XSL stylesheets to `any.xml` (to DocBook) followed by a DocBook XSL stylesheet to generate HTML output, as described in [Getting Started](#).

## From HTML to BoostBook

Most library authors are comfortable with writing HTML documentation. Writing [DocBook](#) documentation (and, by extension, BoostBook documentation) is quite similar to writing HTML, except that BoostBook uses different element names from HTML (see [Table 2, “Converting HTML elements to BoostBook”](#)) and BoostBook XML is a much more rigid format than HTML.

One of the easiest ways to convert HTML documentation into BoostBook documentation is to use [HTML Tidy](#) to transform your HTML into valid XHTML, which will make sure that all elements are properly closed, then apply the transformations in [Table 2, “Converting HTML elements to BoostBook”](#) to the body of the XHTML document. The following command uses HTML Tidy to transform HTML into valid XHTML:

```
tidy -asxhtml input.html > output.xhtml
```

When converting documentation from HTML to BoostBook, note that some redundant information that has to be manually maintained in HTML is automatically generated in BoostBook: for instance, the library categorizations, purpose, and author list described in [the section called “Defining a BoostBook library”](#) are used both in the documentation for the library and to build alphabetical and categorized lists of known libraries; similarly, tables of contents are built automatically from the titles of sections in the BoostBook document.

**Table 2. Converting HTML elements to BoostBook**

HTML	BoostBook
<code>&lt;h1&gt;</code> , <code>&lt;h2&gt;</code> , etc.	<code>&lt;section&gt;</code> , <code>&lt;title&gt;</code> ; See <a href="#">the section called “Sectioning in BoostBook”</a>
<code>&lt;i&gt;</code> , <code>&lt;em&gt;</code>	<code>&lt;emphasis&gt;</code>
<code>&lt;b&gt;</code>	<code>&lt;emphasis role="bold"&gt;</code>
<code>&lt;ol&gt;</code>	<code>&lt;orderedlist&gt;</code>
<code>&lt;ul&gt;</code>	<code>&lt;itemizedlist&gt;</code>
<code>&lt;li&gt;</code>	<code>&lt;listitem&gt;</code>
<code>&lt;pre&gt;</code>	<code>&lt;programlisting&gt;</code>
<code>&lt;code&gt;</code>	<code>&lt;computeroutput&gt;</code> , <code>&lt;code&gt;</code>
<code>&lt;p&gt;</code>	<code>&lt;para&gt;</code> , <code>&lt;simpara&gt;</code>
<code>&lt;a&gt;</code>	<code>&lt;xref&gt;</code> , <code>&lt;link&gt;</code> , <code>&lt;ulink&gt;</code> ; See <a href="#">the section called “Linking in BoostBook”</a>
<code>&lt;table&gt;</code> , <code>&lt;tr&gt;</code> , <code>&lt;th&gt;</code> , <code>&lt;td&gt;</code>	<code>&lt;table&gt;</code> , <code>&lt;informaltable&gt;</code> , <code>&lt;tgroup&gt;</code> , <code>&lt;thead&gt;</code> , <code>&lt;tfoot&gt;</code> , <code>&lt;tbody&gt;</code> , <code>&lt;row&gt;</code> , <code>&lt;entry&gt;</code> , <code>&lt;entrytbl&gt;</code> ; BoostBook tables are equivalent to DocBook tables, for which there is a good <a href="#">tutorial here</a>

# Sectioning in BoostBook

"Sectioning" refers to organization of a document into separate sections, each with a title, some text, and possibly subsections. Each section is described in BoostBook via a `<section>` element. An introduction section may look like this:

```
<section id="any.intro">
  <title>Introduction</title>

  <para>Introduction to a library...</para>

  <section>
    <title>A Subsection</title>
    <para>Subsection information...</para>
  </section>
</section>
```

The `<section>` element contains all information that should logically be grouped within that section. The title of the section is placed within the `<title>` element, and any paragraphs, programs, lists, tables, or subsections can occur within the section. The `id` attribute of the `<section>` element gives a unique ID to each section, so that it may later be identified for linking. It is suggested that all IDs start with the short name of a library followed by a period, so that IDs do not conflict between libraries.

---

# Bringing Together a BoostBook Document

## Linking in BoostBook

How one links to another element in BoostBook depends greatly on the nature of the element linked and how the link should appear. There are three general linking elements: `<xref>`, `<link>`, and `<ulink>`. Additionally, there are linking elements for referencing specific types of entities, such as classes (`<classname>`), functions (`<functionname>`), or libraries (`<libraryname>`).

The `<xref>` element references elements that have an `id` attribute and a title. The actual link text is composed from title and type of the element referenced. To link to a particular ID, create an `<xref>` element with the `linkend` attribute set to the ID of the intended target. For instance, this section's ID is `boostbook.linking`, so we create a reference to it with `<xref linkend="boostbook.linking"/>`, which will look like this in the text: [the section called "Linking in BoostBook"](#).

The `<link>` element references an ID in the same way as `<xref>`, except that `<link>` does not generate any text for the link, so text must be supplied within the element. For instance, we can again link to this chapter but this time specify our own text with `<link linkend="boostbook.linking">like this</link>`. This markup will result in a link to this chapter that looks [like this](#).

The `<ulink>` element references a URL that is outside of the DocBook document. The `url` attribute contains the URL to link to, and the element data provides the link text. For instance, we can link to the Boost web site with `<ulink url="http://www.boost.org">Boost</ulink>`, which appears in the document like this: [Boost](http://www.boost.org).

In BoostBook, `<ulink>` supports a custom url schema for linking to files within the boost distribution. This is formed by setting the `url` attribute to `boost:` followed by the file's path. For example, we can link to the flyweight library with `<ulink url="boost:/libs/flyweight/index.html">Boost.Flyweight</ulink>`, which will appear like this: [Boost.Flyweight](#). This schema is only supported for BoostBook `<ulink>` elements. It isn't available for any other elements or in Docbook.

The `<classname>`, `<functionname>`, `<methodname>`, and `<libraryname>` link to classes, functions, methods, and libraries, respectively. The text of each element gives both the name of the element to link to and the link text. For instance, we can link to the Function library with `<libraryname>Function</libraryname>`, which results in the following: [Function](#). In cases where the displayed text is different from the actual name, the `alt` attribute can be specified. For instance, the following XML element references the `boost::function` class template but displays the text `function`: `<classname alt="boost::function">function</classname>`.

---

# Reference

## Elements:

- Element `access` - Declares an access specification for class members
- Element `boostbook` - Defines a BoostBook book
- Element `class` - Declares a class or class template
- Element `class-specialization` - A specialization (partial or full) of a class template
- Element `code` - Mimics the `code` tag in HTML
- Element `compile-fail-test` - A testcase that should fail to compile
- Element `compile-test` - A testcase that should compile correctly
- Element `complexity` - The time/space/etc. complexity of a function
- Element `constructor` - Declares a constructor of the enclosing class
- Element `copy-assignment` - Declares a copy-assignment operator
- Element `data-member` - Declares a data member of a class
- Element `default` - The default value of a function or template parameter
- Element `description` - Detailed description of a construct
- Element `destructor` - Declares a destructor for the enclosing class
- Element `effects` - Declares the side effects of a function
- Element `enum` - Declares an enumeration type
- Element `enumname` - References an enumeration type with the given name
- Element `enumvalue` - A single value of an enumeration
- Element `free-function-group` - A set of functions that are grouped together under one name
- Element `function` - Declares a function
- Element `functionname` - References a function with the given name
- Element `globalname` - References a global with the given name
- Element `header` - Declares a C++ header with the given name
- Element `headername` - References a C++ header with the given name
- Element `if-fails` - What it means when a testcase fails
- Element `inherit` - Declares a base class of the enclosing class or struct
- Element `lib` - A library dependency
- Element `library` - Top-level element for a library
- Element `library-reference` - Declares the reference material for a library
- Element `librarycategory` - Declares that the enclosing library is in this category
- Element `librarycategorydef` - Defines a new library category
- Element `librarycategorylist` - Categorized listing of libraries
- Element `libraryinfo` - Provides information about a library
- Element `librarylist` - Placeholder for an alphabetical list of libraries
- Element `libraryname` - References a library of the given name
- Element `librarypurpose` - Describes in one short sentence or phrase the purpose of a library
- Element `link-fail-test` - Declares a test that should compile but fail to link
- Element `link-test` - Declares a test that should compile and link
- Element `macroname` - References a macro with the given name
- Element `method` - Declares a method, i.e., a member function
- Element `method-group` - A set of methods that are grouped together under one name
- Element `namespace` - Declares a namespace
- Element `notes` - Non-normative notes about a function's semantics
- Element `overloaded-function` - An overloaded function
- Element `overloaded-method` - An overloaded method
- Element `parameter` - A function parameter
- Element `paramtype` - The type of a function parameter
- Element `postconditions` - Conditions that must hold after the function returns
- Element `precondition` - Conditions that must be met prior to executing a function
- Element `programlisting` - A sample of program code
- Element `purpose` - A short description of an entity's use
- Element `rationale` - Describes the rationale for a particular function's design
- Element `requirement` - A requirement/property in the Jamfile for a testcase

- `Element requires` - Declares the requirements of a function
- `Element returns` - Description of the return value of a function
- `Element run-fail-test` - A testcase that should compile and link, but fail on execution
- `Element run-test` - A testcase that should compile, link, and execute
- `Element signature` - One signature of an overloaded function or method
- `Element snippet` - Pulls in a code snippet from a `programlisting` element
- `Element source` - Defines source code for a test
- `Element specialization` - Defines the specialization arguments for a class specialization
- `Element static-constant` - Declares a static constant, e.g., `const int foo = 5;`
- `Element struct` - Declares a C++ struct
- `Element struct-specialization` - A specialization (full or partial) of a struct template
- `Element template` - Declares the template parameters of a class or function
- `Element template-arg` - A template argument in a specialization
- `Element template-nontype-parameter` - A nontype template parameter
- `Element template-type-parameter` - Declares a template type parameter
- `Element template-varargs` - Declares a variable-length list of template parameters
- `Element testsuite` - Describes a library testsuite
- `Element throws` - Description of the exceptions thrown by a function
- `Element type` - The type of an element or return type of a function
- `Element typedef` - Declares a typedef
- `Element union` - Declares a C++ union or union template
- `Element union-specialization` - A specialization (full or partial) of a union template
- `Element using-class` - Injects the method and function names of a class into the local scope
- `Element using-namespace` - Injects the declared names from a namespace into the local scope

# BoostBook element class-specialization

class-specialization — A specialization (partial or full) of a class template

## Synopsis

class-specialization ::= ([template?](#), [specialization?](#), [inherit?](#), [purpose?](#), [description?](#), ([access](#) | [static-constant](#) | [typedef](#) | [enum](#) | [copy-assignment](#) | [constructor](#) | [destructor](#) | [method-group](#) | [free-function-group](#) | [function](#) | [method](#) | [overloaded-function](#) | [overloaded-method](#) | [data-member](#) | [class](#) | [class-specialization](#) | [struct](#) | [struct-specialization](#) | [union](#) | [union-specialization](#))\*)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element link-test

link-test — Declares a test that should compile and link

## Synopsis

link-test ::= ([source](#)\*, [lib](#)\*, [requirement](#)\*, [purpose](#), [if-fails](#)?)

## Attributes

Name	Type	Value	Purpose
filename	#REQUIRED	CDATA	The name of the file associated with this element
name	#IMPLIED	CDATA	The name of the element being declared to referenced

# BoostBook element link-fail-test

link-fail-test — Declares a test that should compile but fail to link

## Synopsis

link-fail-test ::= ([source\\*](#), [lib\\*](#), [requirement\\*](#), [purpose](#), [if-fails?](#))

## Attributes

Name	Type	Value	Purpose
filename	#REQUIRED	CDATA	The name of the file associated with this element
name	#IMPLIED	CDATA	The name of the element being declared to referenced



# BoostBook element typedef

typedef — Declares a typedef

## Synopsis

typedef ::= ([type](#), [purpose?](#), [description?](#))

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element static-constant

static-constant — Declares a static constant, e.g., `const int foo = 5;`.

## Synopsis

static-constant ::= (type, default, purpose?, description?)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element code

code — Mimics the `code` tag in HTML

## Synopsis

code ::= (ANY)

## Description

Text within a `code` tag is generally typeset in a different, monospaced font so that it stands out as code. The `code` tag in BoostBook is transformed directly into the `computeroutput` tag in DocBook.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element destructor

destructor — Declares a destructor for the enclosing class

## Synopsis

destructor ::= ([purpose?](#), [description?](#), [requires?](#), [effects?](#), [postconditions?](#), [returns?](#), [throws?](#), [complexity?](#), [notes?](#), [rationale?](#))

## Description

General documentation on functions in BoostBook is provided in the [function](#) element documentation.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
specifiers	#IMPLIED	CDATA	The specifiers for this function, e.g., inline, static, etc.
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element template-type-parameter

template-type-parameter — Declares a template type parameter

## Synopsis

template-type-parameter ::= (default?, purpose?)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
pack	#IMPLIED	CDATA	Set to '1' if the parameter is a parameter pack.
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element description

description — Detailed description of a construct

## Synopsis

description ::= (ANY)

## Description

Although the context model for this element is *ANY*, detailed descriptions should contain structured DocBook elements that occur within sections, e.g., paragraphs (*para*, *simpara*), lists (*orderedlist*, *itemizedlist*), tables (*informaltable*, *table*), etc.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element librarylist

librarylist — Placeholder for an alphabetical list of libraries

## Synopsis

librarylist ::= EMPTY

## Description

Developers aren't generally expected to use this element. Its existence is mainly as a placeholder in `boost.xml` for the alphabetical list of libraries.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element library-reference

library-reference — Declares the reference material for a library

## Synopsis

library-reference ::= (title?, section\*, ([header](#) | [library-reference](#))\*)

## Description

Reference documentation for a library is contained with a `<library-reference>` element. The `<library-reference>` element has no attributes, and contains as children only `<header>` elements.

The `<header>` element defines a C++ header file. Within each C++ header file lie the definitions of C++ constructs to be documented. The `name` attribute of the `<header>` element gives the name of the header, as one would specify when including the header. For instance, the `<library-reference>` for the Any library may look like this:

```
<library-reference>
  <header name="boost/any.hpp">
    <!-- C++ constructs in this header -->
  </header>
</library-reference>
```

If the Any library contained multiple headers, we would list them all as children of the `<library-reference>` element.

`library-reference` elements can be nested, so that reference material can be divided into separate sections that each contain different headers.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes



# BoostBook element boostbook

boostbook — Defines a BoostBook book

## Synopsis

boostbook ::= (title, (chapter| [library](#))\*)

## Description

This element is the topmost level defined by `boost.xml` for all Boost documentation. It will not generally be used by developers.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element union

union — Declares a C++ union or union template

## Synopsis

union ::= (template?, inherit\*, purpose?, description?, (access|static-constant|typedef|enum|copy-assignment|constructor|destructor|method-group|free-function-group|function|method|overloaded-function|overloaded-method|data-member|class|class-specialization|struct|struct-specialization|union|union-specialization)\*)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element inherit

inherit — Declares a base class of the enclosing class or struct

## Synopsis

inherit ::= (type, purpose?)

## Description

This element contains the type of the class inherited.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
access	#IMPLIED	CDATA	The access specifier ("public", "private", or "protected") of the inheritance.
id	#IMPLIED	CDATA	A global identifier for this element
pack	#IMPLIED	CDATA	Set to '1' if this is a pack expansion.
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element template-varargs

template-varargs — Declares a variable-length list of template parameters

## Synopsis

template-varargs ::= EMPTY

## Description

Variable-length template parameter lists are not allowed in C++, but because they are sometimes needed in documentation they are allowed in BoostBook. This element generally expands to "... " and can be used anywhere any other template parameter can be used.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element source

source — Defines source code for a test

## Synopsis

source ::= (#PCDATA| [snippet](#))\*

## Description

This element will contain the source code for a testcase that will be generated from the documentation. To reduce the amount of escaping in the text, it is recommended to use CDATA sections, which look like this:

```
<![CDATA[  
<your program text here: no escaping needed!>  
]]>
```

In addition to CDATA sections, code snippets can be pulled in from `programlisting` elements using the [snippet](#) element.

# BoostBook element function

function — Declares a function

## Synopsis

function ::= (template?, type, parameter\*, purpose?, description?, requires?, effects?, postconditions?, returns?, throws?, complexity?, notes?, rationale?)

## Description

BoostBook functions are documented by specifying the function's interface (e.g., its C++ signature) and its behavior. Constructors, destructors, member functions, and free functions all use the same documentation method, although the top-level tags differ.

The behavior of functions in BoostBook is documenting using a style similar to that of the C++ standard, with clauses describing the requirements, effects, postconditions, exception behavior, and return values of functions.

The following example illustrates some constructors and a destructor for `boost::any`. Note that one of the constructors takes a single parameter whose name is "other" and whose type, `const any&` is contained in the `<paramtype>` element; any number of parameters may be specified in this way.

```
<class name="any">
  <constructor>
    <postconditions><para><this->empty()</para></postconditions>
  </constructor>

  <constructor>
    <parameter name="other">
      <paramtype>const <classname>any</classname>&&</paramtype>
    </parameter>

    <effects>
      <simpara>Copy constructor that copies
        content of <code>other</code> into the new instance,
        so that any content is equivalent in both type and value to the
        content of <code>other</code>, or empty if
        <code>other</code> is
        empty.
      </simpara>
    </effects>

    <throws>
      <simpara>May fail with a
        <classname>std::bad_alloc</classname> exception or any
        exceptions arising from the copy constructor of the
        contained type.
      </simpara>
    </throws>
  </constructor>

  <destructor>
    <effects><simpara>Releases any and all resources used in
      management of instance.</simpara></effects>

    <throws><simpara>Nothing.</simpara></throws>
  </destructor>
</class>
```

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
specifiers	#IMPLIED	CDATA	The specifiers for this function, e.g., inline, static, etc.
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element macroname

macroname

## Synopsis

macroname ::= (#PCDATA)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
alt	#IMPLIED	CDATA	
id	#IMPLIED	CDATA	A global identifier for this element



# BoostBook element postconditions

postconditions — Conditions that must hold after the function returns

## Synopsis

postconditions ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element compile-test

compile-test — A testcase that should compile correctly

## Synopsis

compile-test ::= (source\*, lib\*, requirement\*, purpose, if-fails?)

## Attributes

Name	Type	Value	Purpose
filename	#REQUIRED	CDATA	The name of the file associated with this element
name	#IMPLIED	CDATA	The name of the element being declared to referenced

# BoostBook element method

method — Declares a method, i.e., a member function

## Synopsis

method ::= ([template?](#), [type](#), [parameter\\*](#), [purpose?](#), [description?](#), [requires?](#), [effects?](#), [postconditions?](#), [returns?](#), [throws?](#), [complexity?](#), [notes?](#), [rationale?](#))

## Description

General documentation on functions in BoostBook is provided in the [function](#) element documentation.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
cv	#IMPLIED	CDATA	cv-qualifiers for this method, e.g., const volatile
specifiers	#IMPLIED	CDATA	The specifiers for this function, e.g., inline, static, etc.
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element snippet

snippet — Pulls in a code snippet from a `programlisting` element

## Synopsis

snippet ::= EMPTY

## Attributes

Name	Type	Value	Purpose
name	#REQUIRED	CDATA	The name of the <code>programlisting</code> element to insert

# BoostBook element constructor

constructor — Declares a constructor of the enclosing class

## Synopsis

constructor ::= ([template?](#), [parameter\\*](#), [purpose?](#), [description?](#), [requires?](#), [effects?](#), [postconditions?](#), [returns?](#), [throws?](#), [complexity?](#), [notes?](#), [rationale?](#))

## Description

General documentation on functions in BoostBook is provided in the [function](#) element documentation.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
specifiers	#IMPLIED	CDATA	The specifiers for this function, e.g., inline, static, etc.
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element namespace

namespace — Declares a namespace

## Synopsis

namespace ::= ([class](#)| [class-specialization](#)| [struct](#)| [struct-specialization](#)| [union](#)| [union-specialization](#)| [typedef](#)| [enum](#)| [free-function-group](#)| [function](#)| [overloaded-function](#)| [namespace](#))\*

## Description

BoostBook namespaces are declared via the <namespace> element. As in C++, namespaces can be nested and contain other C++ constructs, such as classes or functions. The name attribute of a <namespace> element gives the namespace name (e.g., "boost"). The Any library is defined entirely within namespace boost by:

```
<library-reference>
  <header name="boost/any.hpp">
    <namespace name="boost">
      <!-- C++ constructs in the boost namespace -->
    </namespace>
  </header>
</library-reference>
```

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element if-fails

if-fails — What it means when a testcase fails

## Synopsis

if-fails ::= (ANY)

## Description

Describes to the user the effect a certain failing testcase will have on the usefulness of a library. This field is useful in cases where a failed testcase does not mean that the library won't be useful, but may mean that certain library features will not be available.

# BoostBook element headername

headername

## Synopsis

headername ::= (#PCDATA)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
alt	#IMPLIED	CDATA	
id	#IMPLIED	CDATA	A global identifier for this element



# BoostBook element free-function-group

free-function-group — A set of functions that are grouped together under one name

## Synopsis

free-function-group ::= ([function](#) | [overloaded-function](#))\*

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element `functionname`

`functionname` — References a function with the given name

## Synopsis

`functionname ::= (#PCDATA)`

## Description

If a function (or overloaded function) with the given, possibly-qualified name is found, this generates a link to that function. Lookups obey currently-active `using-class` and `using-namespace` directives to aid in the search, along with searching within the current scope.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
alt	#IMPLIED	CDATA	
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element librarycategory

librarycategory — Declares that the enclosing library is in this category

## Synopsis

librarycategory ::= (#PCDATA)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element notes

notes — Non-normative notes about a function's semantics

## Synopsis

notes ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element data-member

data-member — Declares a data member of a class

## Synopsis

data-member ::= (type, purpose?, description?)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
specifiers	#IMPLIED	CDATA	The specifiers for this function, e.g., inline, static, etc.
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element specialization

specialization — Defines the specialization arguments for a class specialization

## Synopsis

specialization ::= ([template-arg](#))\*

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element union-specialization

union-specialization — A specialization (full or partial) of a union template

## Synopsis

union-specialization ::= ([template?](#), [specialization?](#), [inherit?](#), [purpose?](#), [description?](#), ([access](#)|[static-constant](#)|[typedef](#)|[enum](#)|[copy-assignment](#)|[constructor](#)|[destructor](#)|[method-group](#)|[free-function-group](#)|[function](#)|[method](#)|[overloaded-function](#)|[overloaded-method](#)|[data-member](#)|[class](#)|[class-specialization](#)|[struct](#)|[struct-specialization](#)|[union](#)|[union-specialization](#)\*)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element throws

throws — Description of the exceptions thrown by a function

## Synopsis

throws ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes



# BoostBook element template-arg

template-arg — A template argument in a specialization

## Synopsis

template-arg ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
pack	#IMPLIED	CDATA	Set to '1' if this is a pack expansion.
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element globalname

globalname

## Synopsis

globalname ::= (#PCDATA)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
alt	#IMPLIED	CDATA	
id	#IMPLIED	CDATA	A global identifier for this element

# BoostBook element method-group

method-group — A set of methods that are grouped together under one name

## Synopsis

method-group ::= ([method](#) | [overloaded-method](#))\*

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element requirement

requirement — A requirement/property in the Jamfile for a testcase

## Synopsis

requirement ::= (#PCDATA)

## Description

A requirement is part of the dependencies of a target in a Jamfile. The `name` attribute of a requirement element gives the name of the Boost.Build feature and the content of the requirement gives the value of that feature. A requirement such as `<includes>foo.hpp` would be encoded as `<requirement name="includes">foo.hpp</requirement>`.

## Attributes

Name	Type	Value	Purpose
name	#REQUIRED	CDATA	The name of the element being declared to referenced

# BoostBook element precondition

precondition — Conditions that must be met prior to executing a function

## Synopsis

precondition ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element paramtype

paramtype — The type of a function parameter

## Synopsis

paramtype ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element using-class

using-class — Injects the method and function names of a class into the local scope

## Synopsis

using-class ::= EMPTY

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element run-test

run-test — A testcase that should compile, link, and execute

## Synopsis

run-test ::= ([source](#)\*, [lib](#)\*, [requirement](#)\*, [purpose](#), [if-fails](#)?)

## Attributes

Name	Type	Value	Purpose
filename	#REQUIRED	CDATA	The name of the file associated with this element
name	#IMPLIED	CDATA	The name of the element being declared to referenced



# BoostBook element librarypurpose

librarypurpose — Describes in one short sentence or phrase the purpose of a library

## Synopsis

librarypurpose ::= (#PCDATA| [code](#)| [ulink](#)| [functionname](#)| [methodname](#)| [classname](#)| [macroname](#)| [headername](#)| [enumname](#)| [globalname](#))\*

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element copy-assignment

copy-assignment — Declares a copy-assignment operator

## Synopsis

copy-assignment ::= (template?, type?, parameter\*, purpose?, description?, requires?, effects?, postconditions?, returns?, throws?, complexity?, notes?, rationale?)

## Description

The return type of the copy-assignment operator does not need to be specified. If left unspecified, it will default to an unqualified reference to the enclosing class type.

General documentation on functions in BoostBook is provided in the [function](#) element documentation.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
cv	#IMPLIED	CDATA	cv-qualifiers for this method, e.g., const volatile
specifiers	#IMPLIED	CDATA	The specifiers for this function, e.g., inline, static, etc.
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element run-fail-test

run-fail-test — A testcase that should compile and link, but fail on execution

## Synopsis

run-fail-test ::= ([source](#)\*, [lib](#)\*, [requirement](#)\*, [purpose](#), [if-fails](#)?)

## Attributes

Name	Type	Value	Purpose
filename	#REQUIRED	CDATA	The name of the file associated with this element
name	#IMPLIED	CDATA	The name of the element being declared to referenced

# BoostBook element template

template — Declares the template parameters of a class or function

## Synopsis

template ::= ([template-type-parameter](#) | [template-nontype-parameter](#) | [template-varargs](#))\*

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element compile-fail-test

compile-fail-test — A testcase that should fail to compile

## Synopsis

compile-fail-test ::= ([source](#)\*, [lib](#)\*, [requirement](#)\*, [purpose](#), [if-fails](#)?)

## Attributes

Name	Type	Value	Purpose
filename	#REQUIRED	CDATA	The name of the file associated with this element
name	#IMPLIED	CDATA	The name of the element being declared to referenced

# BoostBook element returns

returns — Description of the return value of a function

## Synopsis

returns ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element default

default — The default value of a function or template parameter

## Synopsis

default ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element parameter

parameter — A function parameter

## Synopsis

parameter ::= ([paramtype](#), [default?](#), [description?](#))

## Attributes

Name	Type	Value	Purpose
name	#IMPLIED	CDATA	The name of the element being declared to referenced
pack	#IMPLIED	CDATA	Set to '1' if the parameter is a parameter pack.



# BoostBook element signature

signature — One signature of an overloaded function or method

## Synopsis

signature ::= ([template?](#), [type](#), [parameter\\*](#))

## Description

A signature refers to one declaration of an overloaded function or method. The signature itself has no name, because the name of the overloaded function or method is used. An overloaded function or method will have several signatures that will generally be typeset together.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
cv	#IMPLIED	CDATA	cv-qualifiers for this method, e.g., const volatile
specifiers	#IMPLIED	CDATA	The specifiers for this function, e.g., inline, static, etc.
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element overloaded-function

overloaded-function — An overloaded function

## Synopsis

overloaded-function ::= ([signature\\*](#), [purpose?](#), [description?](#), [requires?](#), [effects?](#), [postconditions?](#), [returns?](#), [throws?](#), [complexity?](#), [notes?](#), [rationale?](#))

## Description

General documentation on functions in BoostBook is provided in the [function](#) element documentation.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element access

access — Declares an access specification for class members

## Synopsis

access ::= ([static-constant](#) | [typedef](#) | [enum](#) | [copy-assignment](#) | [constructor](#) | [destructor](#) | [method-group](#) | [method](#) | [overloaded-method](#) | [data-member](#) | [class](#) | [class-specialization](#) | [struct](#) | [struct-specialization](#) | [union](#) | [union-specialization](#)) +

## Description

The access specifications of class members (public, private, or protected) may be determined by enclosing them in an <access> element.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the access specification, e.g. "public", "private", or "protected".
id	#IMPLIED	CDATA	A global identifier for this element

# BoostBook element class

class — Declares a class or class template

## Synopsis

class ::= ([template?](#), [inherit\\*](#), [purpose?](#), [description?](#), ([access](#)|[static-constant](#)|[typedef](#)|[enum](#)|[copy-assignment](#)|[constructor](#)|[destructor](#)|[method-group](#)|[free-function-group](#)|[function](#)|[method](#)|[overloaded-function](#)|[overloaded-method](#)|[data-member](#)|[class](#)|[class-specialization](#)|[struct](#)|[struct-specialization](#)|[union](#)|[union-specialization](#))\*)

## Description

C++ classes and class templates are described via the `<class>` element. Each class has a name (e.g., "any") given by the `name` attribute, a purpose given by the `<purpose>` element, documentation, and a set of types, functions, base classes, and data members. Here is a minimal definition of the `boost::any` class:

```
<namespace name="boost">
  <class name="any">
    <purpose>
      A class whose instances can hold instances of any type that satisfies
      ValueType requirements.
    </purpose>
  </class>
</namespace>
```

Additional class documentation can be contained in a `<description>` element following the `<purpose>` element. This documentation will be typeset prior to documentation for specific elements in the class (e.g., constructors or methods).

Class inheritance is described via the `<inherit>` element. The `<inherit>` element requires an `access` attribute which must be one of *public*, *protected*, or *private*. The content of the `<inherited>` element in C++ code that names the class inherited, and may contain markup to link to the class. The following description of the class `boost::bad_any_cast` describes public inheritance from the class `std::bad_cast`. It also defines the `<purpose>` element, which contains a short description of the use of the class.

```
<class name="bad_any_cast">
  <inherit access="public"><classname>std::bad_cast</classname></inherit>
  <purpose><para>The exception thrown in the event of a failed
  <functionname>any_cast</functionname> of an
  <classname>any</classname> value.</para></purpose>
</class>
```

Class templates are defined by `<class>` elements with a `<template>` child element at the beginning.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element librarycategorydef

librarycategorydef — Defines a new library category

## Synopsis

librarycategorydef ::= (#PCDATA)

## Description

All library category definitions should be in `doc/src/boost.xml`, and the names of categories must be prefixed with "category:".

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element type

type — The type of an element or return type of a function

## Synopsis

type ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element enumvalue

enumvalue — A single value of an enumeration

## Synopsis

enumvalue ::= (default?, purpose?, description?)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element overloaded-method

overloaded-method — An overloaded method

## Synopsis

overloaded-method ::= ([signature](#)\*, [purpose](#)?, [description](#)?, [requires](#)?, [effects](#)?, [postconditions](#)?, [returns](#)?, [throws](#)?, [complexity](#)?, [notes](#)?, [rationale](#)?)

## Description

General documentation on functions in BoostBook is provided in the [function](#) element documentation.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes



# BoostBook element programlisting

programlisting — A sample of program code

## Synopsis

programlisting ::= (ANY)

## Attributes

Name	Type	Value	Purpose
name	#IMPLIED	CDATA	The name of the element being declared to referenced

# BoostBook element complexity

complexity — The time/space/etc. complexity of a function

## Synopsis

complexity ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element purpose

purpose — A short description of an entity's use

## Synopsis

purpose ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element template-nontype-parameter

template-nontype-parameter — A nontype template parameter

## Synopsis

template-nontype-parameter ::= (type, default?, purpose?)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
pack	#IMPLIED	CDATA	Set to '1' if the parameter is a parameter pack.
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element library

library — Top-level element for a library

## Synopsis

library ::= ([libraryinfo](#), (title, ((section| [library-reference](#)| [testsuite](#)))+)?)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
dirname	#REQUIRED	CDATA	
url	#IMPLIED	CDATA	
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes
html-only	#IMPLIED	CDATA	

# BoostBook element librarycategorylist

librarycategorylist — Categorized listing of libraries

## Synopsis

librarycategorylist ::= ([librarycategorydef](#))\*

## Description

This element is not intended for use by developers, but is used by `doc/src/boost.xml` as a placeholder.

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element using-namespace

using-namespace — Injects the declared names from a namespace into the local scope

## Synopsis

using-namespace ::= EMPTY

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element enumname

enumname

## Synopsis

enumname ::= (#PCDATA)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
alt	#IMPLIED	CDATA	
id	#IMPLIED	CDATA	A global identifier for this element



# BoostBook element struct-specialization

struct-specialization — A specialization (full or partial) of a struct template

## Synopsis

struct-specialization ::= (template?, specialization?, inherit?, purpose?, description?, (access| static-constant| typedef| enum| copy-assignment| constructor| destructor| method-group| free-function-group| function| method| overloaded-function| overloaded-method| data-member| class| class-specialization| struct| struct-specialization| union| union-specialization)\*)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element struct

struct — Declares a C++ struct

## Synopsis

struct ::= (template?, inherit\*, purpose?, description?, (access| static-constant| typedef| enum| copy-assignment| constructor| destructor| method-group| free-function-group| function| method| overloaded-function| overloaded-method| data-member| class| class-specialization| struct| struct-specialization| union| union-specialization)\*)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element lib

lib — A library dependency

## Synopsis

lib ::= (#PCDATA)

## Description

Declares a library dependency on the library named by the content of this element, to be emitted in a Jamfile.

# BoostBook element enum

enum — Declares an enumeration type

## Synopsis

enum ::= (enumvalue\*, purpose?, description?)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element requires

requires — Declares the requirements of a function

## Synopsis

requires ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element effects

effects — Declares the side effects of a function

## Synopsis

effects ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element libraryname

libraryname — References a library of the given name

## Synopsis

libraryname ::= (#PCDATA)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element libraryinfo

libraryinfo — Provides information about a library

## Synopsis

libraryinfo ::= (author+, copyright\*, legalnotice\*, [librarypurpose](#), [librarycategory](#)\*)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes



# BoostBook element testsuite

testsuite — Describes a library testsuite

## Synopsis

testsuite ::= (([compile-test](#) | [link-test](#) | [run-test](#) | [compile-fail-test](#) | [link-fail-test](#) | [run-fail-test](#)) +)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element header

header — Declares a C++ header with the given name

## Synopsis

header ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
name	#REQUIRED	CDATA	The name of the element being declared to referenced
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes

# BoostBook element rationale

rationale — Describes the rationale for a particular function's design

## Synopsis

rationale ::= (ANY)

## Attributes

Name	Type	Value	Purpose
last-revision	#IMPLIED	CDATA	Set to \$Date: 2009-10-10 15:53:46 +0100 (Sat, 10 Oct 2009) \$ to keep "last revised" information in sync with CVS changes
id	#IMPLIED	CDATA	A global identifier for this element
xml:base	#IMPLIED	CDATA	Implementation detail used by XIncludes