

**M3P SCoPE XML Plug-in**  
**1.0.0**

Generated by Doxygen 1.5.5

Wed Dec 17 19:05:24 2008

## Contents

<a href="#">1 Namespace Index</a>	1
<a href="#">2 Data Structure Index</a>	1
<a href="#">3 Data Structure Index</a>	2
<a href="#">4 File Index</a>	2
<a href="#">5 Namespace Documentation</a>	4
<a href="#">6 Data Structure Documentation</a>	4
<a href="#">7 File Documentation</a>	29

## 1 Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">expatmm</a>	4
-------------------------	---

## 2 Data Structure Index

### 2.1 Class Hierarchy

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

<a href="#">expatmm::ExpatXMLParser</a>	4
<a href="#">MetricReader</a>	19
<a href="#">hw_component</a>	8
<a href="#">hw_connection</a>	10
<a href="#">hw_group</a>	11
<a href="#">hw_instance</a>	12
<a href="#">list_elem</a>	14
<a href="#">Metric</a>	14
<a href="#">sw_allocation</a>	23
<a href="#">sw_component</a>	24

<a href="#">sw_instance</a>	25
<a href="#">sw_task</a>	26
<a href="#">xml_basic_info</a>	27
<a href="#">xml_parameter_info</a>	28
<a href="#">xml_scope_simulation_info_t</a>	28

## 3 Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">expatmm::ExpatXMLParser</a>	4
<a href="#">hw_component</a>	8
<a href="#">hw_connection</a>	10
<a href="#">hw_group</a>	11
<a href="#">hw_instance</a>	12
<a href="#">list_elem</a>	14
<a href="#">Metric</a> ( <a href="#">Metric</a> Class which captures the design metric concept )	14
<a href="#">MetricReader</a> ( <a href="#">Metric</a> (expat + <a href="#">expatmm</a> ) Parser Class )	19
<a href="#">sw_allocation</a>	23
<a href="#">sw_component</a>	24
<a href="#">sw_instance</a>	25
<a href="#">sw_task</a>	26
<a href="#">xml_basic_info</a>	27
<a href="#">xml_parameter_info</a>	28
<a href="#">xml_scope_simulation_info_t</a>	28

## 4 File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">expatmm-libdef.h</a>	29
----------------------------------	----

ExpatMM-version.cpp	29
expatmm.h	29
ExpatXMLParser.cpp	30
src/ExpatXMLParser.h	30
expatmm/ExpatXMLParser.h	30
metric.cpp	31
metric.h	31
parse.cpp	32
parse.h	32
uc_create_xml_platform.cpp	32
uc_create_xml_platform.h	36
uc_load_xml.cpp	37
uc_load_xml.h	39
xml_configuration_file.c	41
xml_configuration_file.h	42
xml_hierarchy.c	43
xml_hierarchy.h	48
xml_if.c	51
xml_if.h	56
xml_input.c	61
xml_input.h	70
xml_list.c	78
xml_list.h	79
xml_main.c	80
xml_main.h	81
xml_obtain_metrics.cpp	81
xml_obtain_metrics.h	83

## 5 Namespace Documentation

### 5.1 expatmm Namespace Reference

#### Data Structures

- class [ExpatXMLParser](#)

#### Functions

- std::string [getExpatMMVersion](#) (void)

#### 5.1.1 Function Documentation

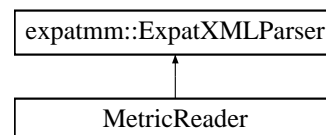
##### 5.1.1.1 std::string EXPATMM\_LIBEXPORT expatmm::getExpatMMVersion (void)

## 6 Data Structure Documentation

### 6.1 expatmm::ExpatXMLParser Class Reference

```
#include <ExpatXMLParser.h>
```

Inheritance diagram for expatmm::ExpatXMLParser::



#### Public Member Functions

- [ExpatXMLParser](#) (void)
- [ExpatXMLParser](#) (size\_t chunk\_size)
- virtual [~ExpatXMLParser](#) (void)
- virtual bool [Parse](#) (void)
- virtual bool [Ready](#) (void)
- virtual XML\_Error [getLastError](#) (void)
- virtual XML\_Status [getStatus](#) (void)
- [ExpatXMLParser](#) (void)
- [ExpatXMLParser](#) (size\_t chunk\_size)
- virtual [~ExpatXMLParser](#) (void)
- virtual bool [Parse](#) (void)
- virtual bool [Ready](#) (void)
- virtual XML\_Error [getLastError](#) (void)
- virtual XML\_Status [getStatus](#) (void)

**Protected Member Functions**

- virtual XML\_Char \* [getBuffer](#) (void)
- virtual size\_t [getBlockSize](#) (void)
- virtual ssize\_t [read\\_block](#) (void)
- virtual void [setReadiness](#) (bool ready)
- virtual void [setStatus](#) (XML\_Status new\_status)
- virtual void [setLastError](#) (XML\_Error new\_last\_error)
- virtual void [StartElement](#) (const XML\_Char \*name, const XML\_Char \*\*atts)
- virtual void [EndElement](#) (const XML\_Char \*name)
- virtual void [CharacterData](#) (const XML\_Char \*s, int len)
- virtual void [ProcessingInstruction](#) (const XML\_Char \*target, const XML\_Char \*data)
- virtual void [CommentData](#) (const XML\_Char \*data)
- virtual void [DefaultHandler](#) (const XML\_Char \*s, int len)
- virtual void [CDataStart](#) (void)
- virtual void [CDataEnd](#) (void)
- virtual XML\_Char \* [getBuffer](#) (void)
- virtual size\_t [getBlockSize](#) (void)
- virtual ssize\_t [read\\_block](#) (void)
- virtual void [setReadiness](#) (bool ready)
- virtual void [setStatus](#) (XML\_Status new\_status)
- virtual void [setLastError](#) (XML\_Error new\_last\_error)
- virtual void [StartElement](#) (const XML\_Char \*name, const XML\_Char \*\*atts)
- virtual void [EndElement](#) (const XML\_Char \*name)
- virtual void [CharacterData](#) (const XML\_Char \*s, int len)
- virtual void [ProcessingInstruction](#) (const XML\_Char \*target, const XML\_Char \*data)
- virtual void [CommentData](#) (const XML\_Char \*data)
- virtual void [DefaultHandler](#) (const XML\_Char \*s, int len)
- virtual void [CDataStart](#) (void)
- virtual void [CDataEnd](#) (void)

**6.1.1 Constructor & Destructor Documentation****6.1.1.1 ExpatXMLParser::ExpatXMLParser (void)****6.1.1.2 ExpatXMLParser::ExpatXMLParser (size\_t *chunk\_size*)****6.1.1.3 ExpatXMLParser::~ExpatXMLParser (void)** [virtual]**6.1.1.4 expatmm::ExpatXMLParser::ExpatXMLParser (void)****6.1.1.5 expatmm::ExpatXMLParser::ExpatXMLParser (size\_t *chunk\_size*)****6.1.1.6 virtual expatmm::ExpatXMLParser::~ExpatXMLParser (void)** [virtual]

## 6.1.2 Member Function Documentation

**6.1.2.1** virtual XML\_Char\* expatmm::ExpatXMLParser::getBuffer (void) [inline, protected, virtual]

**6.1.2.2** virtual size\_t expatmm::ExpatXMLParser::getBlockSize (void) [inline, protected, virtual]

**6.1.2.3** ssize\_t ExpatXMLParser::read\_block (void) [protected, virtual]

Reimplemented in [MetricReader](#).

**6.1.2.4** virtual void expatmm::ExpatXMLParser::setReadiness (bool ready) [inline, protected, virtual]

**6.1.2.5** virtual void expatmm::ExpatXMLParser::setStatus (XML\_Status new\_status) [inline, protected, virtual]

**6.1.2.6** virtual void expatmm::ExpatXMLParser::setLastError (XML\_Error new\_last\_error) [inline, protected, virtual]

**6.1.2.7** void ExpatXMLParser::StartElement (const XML\_Char \* name, const XML\_Char \*\* atts) [protected, virtual]

Reimplemented in [MetricReader](#).

**6.1.2.8** void ExpatXMLParser::EndElement (const XML\_Char \* name) [protected, virtual]

Reimplemented in [MetricReader](#).

**6.1.2.9** void ExpatXMLParser::CharacterData (const XML\_Char \* s, int len) [protected, virtual]

**6.1.2.10** void ExpatXMLParser::ProcessingInstruction (const XML\_Char \* target, const XML\_Char \* data) [protected, virtual]

**6.1.2.11** void ExpatXMLParser::CommentData (const XML\_Char \* data) [protected, virtual]

**6.1.2.12** void ExpatXMLParser::DefaultHandler (const XML\_Char \* s, int len) [protected, virtual]

**6.1.2.13** void ExpatXMLParser::CDataStart (void) [protected, virtual]

**6.1.2.14** void ExpatXMLParser::CDataEnd (void) [protected, virtual]

**6.1.2.15** `bool ExpatXMLParser::Parse (void)` [virtual]

**6.1.2.16** `virtual bool expatmm::ExpatXMLParser::Ready (void)` [inline, virtual]

**6.1.2.17** `virtual XML_Error expatmm::ExpatXMLParser::getLastError (void)` [inline, virtual]

**6.1.2.18** `virtual XML_Status expatmm::ExpatXMLParser::getStatus (void)` [inline, virtual]

**6.1.2.19** `virtual XML_Char* expatmm::ExpatXMLParser::getBuffer (void)` [inline, protected, virtual]

**6.1.2.20** `virtual size_t expatmm::ExpatXMLParser::getBlockSize (void)` [inline, protected, virtual]

**6.1.2.21** `virtual ssize_t expatmm::ExpatXMLParser::read_block (void)` [protected, virtual]

Reimplemented in [MetricReader](#).

**6.1.2.22** `virtual void expatmm::ExpatXMLParser::setReadiness (bool ready)` [inline, protected, virtual]

**6.1.2.23** `virtual void expatmm::ExpatXMLParser::setStatus (XML_Status new_status)` [inline, protected, virtual]

**6.1.2.24** `virtual void expatmm::ExpatXMLParser::setLastError (XML_Error new_last_error)` [inline, protected, virtual]

**6.1.2.25** `virtual void expatmm::ExpatXMLParser::StartElement (const XML_Char * name, const XML_Char ** atts)` [protected, virtual]

Reimplemented in [MetricReader](#).

**6.1.2.26** `virtual void expatmm::ExpatXMLParser::EndElement (const XML_Char * name)` [protected, virtual]

Reimplemented in [MetricReader](#).

**6.1.2.27** `virtual void expatmm::ExpatXMLParser::CharacterData (const XML_Char * s, int len)` [protected, virtual]

**6.1.2.28** `virtual void expatmm::ExpatXMLParser::ProcessingInstruction (const XML_Char * target, const XML_Char * data)` [protected, virtual]



**6.1.2.29** virtual void expatmm::ExpatXMLParser::CommentData (const XML\_Char \* *data*)  
[protected, virtual]

**6.1.2.30** virtual void expatmm::ExpatXMLParser::DefaultHandler (const XML\_Char \* *s*, int *len*)  
[protected, virtual]

**6.1.2.31** virtual void expatmm::ExpatXMLParser::CDataStart (void) [protected, virtual]

**6.1.2.32** virtual void expatmm::ExpatXMLParser::CDataEnd (void) [protected, virtual]

**6.1.2.33** virtual bool expatmm::ExpatXMLParser::Parse (void) [virtual]

**6.1.2.34** virtual bool expatmm::ExpatXMLParser::Ready (void) [inline, virtual]

**6.1.2.35** virtual XML\_Error expatmm::ExpatXMLParser::getLastError (void) [inline, virtual]

**6.1.2.36** virtual XML\_Status expatmm::ExpatXMLParser::getStatus (void) [inline, virtual]

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

- [src/ExpatXMLParser.h](#)
- [expatmm/ExpatXMLParser.h](#)
- [ExpatXMLParser.cpp](#)

## 6.2 hw\_component Struct Reference

```
#include <xml_if.h>
```

### Data Fields

- char \* [name](#)
- char \* [path](#)
- char \* [type](#)
- char \* [class\\_name](#)
- char \* [activation\\_type](#)
- char \* [freq](#)
- char \* [mem\\_size](#)
- char \* [type\\_specific\\_1](#)
- char \* [type\\_specific\\_2](#)
- char \* [area](#)
- char \* [mean\\_power](#)
- char \* [read\\_energy](#)
- char \* [read\\_size\\_energy](#)

- char \* [write\\_energy](#)
- char \* [write\\_size\\_energy](#)
- char \* [latency](#)
- char \* [width](#)

### 6.2.1 Field Documentation

**6.2.1.1** char\* hw\_component::name

**6.2.1.2** char\* hw\_component::path

**6.2.1.3** char\* hw\_component::type

**6.2.1.4** char\* hw\_component::class\_name

**6.2.1.5** char\* hw\_component::activation\_type

**6.2.1.6** char\* hw\_component::freq

**6.2.1.7** char\* hw\_component::mem\_size

**6.2.1.8** char\* hw\_component::type\_specific\_1

**6.2.1.9** char\* hw\_component::type\_specific\_2

**6.2.1.10** char\* hw\_component::area

**6.2.1.11** char\* hw\_component::mean\_power

**6.2.1.12** char\* hw\_component::read\_energy

**6.2.1.13** char\* hw\_component::read\_size\_energy

**6.2.1.14** char\* hw\_component::write\_energy

**6.2.1.15** char\* hw\_component::write\_size\_energy

**6.2.1.16** char\* hw\_component::latency

#### 6.2.1.17 char\* hw\_component::width

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

- [xml\\_if.h](#)

### 6.3 hw\_connection Struct Reference

```
#include <xml_if.h>
```

#### Data Fields

- char \* [name](#)
- char \* [path](#)
- char \* [activation\\_type](#)
- char \* [speed](#)
- char \* [latency](#)
- char \* [mem\\_size](#)
- char \* [type\\_specific\\_1](#)
- char \* [type\\_specific\\_2](#)
- char \* [start\\_addr](#)
- char \* [port](#)
- char \* [irq](#)
- char \* [rec\\_irq](#)
- char \* [local\\_id](#)
- char \* [instance\\_name](#)
- struct [hw\\_instance](#) \* [instance](#)
- int [offset](#)

#### 6.3.1 Field Documentation

##### 6.3.1.1 char\* hw\_connection::name

##### 6.3.1.2 char\* hw\_connection::path

##### 6.3.1.3 char\* hw\_connection::activation\_type

##### 6.3.1.4 char\* hw\_connection::speed

##### 6.3.1.5 char\* hw\_connection::latency

##### 6.3.1.6 char\* hw\_connection::mem\_size

##### 6.3.1.7 char\* hw\_connection::type\_specific\_1

##### 6.3.1.8 char\* hw\_connection::type\_specific\_2

**6.3.1.9** char\* hw\_connection::start\_addr

**6.3.1.10** char\* hw\_connection::port

**6.3.1.11** char\* hw\_connection::irq

**6.3.1.12** char\* hw\_connection::rec\_irq

**6.3.1.13** char\* hw\_connection::local\_id

**6.3.1.14** char\* hw\_connection::instance\_name

**6.3.1.15** struct hw\_instance\* hw\_connection::instance [read]

**6.3.1.16** int hw\_connection::offset

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

- [xml\\_if.h](#)

## 6.4 hw\_group Struct Reference

```
#include <xml_if.h>
```

### Data Fields

- char \* [name](#)
- char \* [path](#)
- struct [list\\_elem](#) \* [hw\\_names](#)
- struct [list\\_elem](#) \* [list](#)

### 6.4.1 Field Documentation

**6.4.1.1** char\* hw\_group::name

**6.4.1.2** char\* hw\_group::path

**6.4.1.3** struct list\_elem\* hw\_group::hw\_names [read]

**6.4.1.4** struct list\_elem\* hw\_group::list [read]

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

- [xml\\_if.h](#)

## 6.5 hw\_instance Struct Reference

```
#include <xml_if.h>
```

### Data Fields

- char \* [name](#)
- char \* [path](#)
- char \* [type](#)
- char \* [activation\\_type](#)
- char \* [class\\_name](#)
- char \* [latency](#)
- char \* [freq](#)
- char \* [mem\\_size](#)
- char \* [type\\_specific\\_1](#)
- char \* [type\\_specific\\_2](#)
- char \* [start\\_addr](#)
- char \* [irq](#)
- char \* [local\\_id](#)
- char \* [area](#)
- char \* [mean\\_power](#)
- char \* [read\\_energy](#)
- char \* [read\\_size\\_energy](#)
- char \* [write\\_energy](#)
- char \* [write\\_size\\_energy](#)
- char \* [width](#)
- char \* [component\\_name](#)
- struct [hw\\_component](#) \* [component](#)
- int [offset](#)
- struct [list\\_elem](#) \* [contain](#)
- struct [list\\_elem](#) \* [connections](#)
- void \* [scope\\_data](#)

### 6.5.1 Field Documentation

#### 6.5.1.1 char\* hw\_instance::name

#### 6.5.1.2 char\* hw\_instance::path

#### 6.5.1.3 char\* hw\_instance::type

#### 6.5.1.4 char\* hw\_instance::activation\_type

#### 6.5.1.5 char\* hw\_instance::class\_name

#### 6.5.1.6 char\* hw\_instance::latency

- 6.5.1.7 char\* hw\_instance::freq
- 6.5.1.8 char\* hw\_instance::mem\_size
- 6.5.1.9 char\* hw\_instance::type\_specific\_1
- 6.5.1.10 char\* hw\_instance::type\_specific\_2
- 6.5.1.11 char\* hw\_instance::start\_addr
- 6.5.1.12 char\* hw\_instance::irq
- 6.5.1.13 char\* hw\_instance::local\_id
- 6.5.1.14 char\* hw\_instance::area
- 6.5.1.15 char\* hw\_instance::mean\_power
- 6.5.1.16 char\* hw\_instance::read\_energy
- 6.5.1.17 char\* hw\_instance::read\_size\_energy
- 6.5.1.18 char\* hw\_instance::write\_energy
- 6.5.1.19 char\* hw\_instance::write\_size\_energy
- 6.5.1.20 char\* hw\_instance::width
- 6.5.1.21 char\* hw\_instance::component\_name
- 6.5.1.22 struct hw\_component\* hw\_instance::component [read]
- 6.5.1.23 int hw\_instance::offset
- 6.5.1.24 struct list\_elem\* hw\_instance::contain [read]
- 6.5.1.25 struct list\_elem\* hw\_instance::connections [read]

#### 6.5.1.26 void\* hw\_instance::scope\_data

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

- [xml\\_if.h](#)

## 6.6 list\_elem Struct Reference

```
#include <xml_list.h>
```

### Data Fields

- struct [list\\_elem](#) \* [next](#)
- void \* [elem](#)

### 6.6.1 Field Documentation

#### 6.6.1.1 struct list\_elem\* list\_elem::next [read]

#### 6.6.1.2 void\* list\_elem::elem

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

- [xml\\_list.h](#)

## 6.7 Metric Class Reference

[Metric](#) Class which captures the design metric concept.

```
#include <metric.h>
```

### Public Member Functions

- [Metric](#) ()  
*Metric default constructor.*
- [Metric](#) (const string &name, const string &rtype, const string &runit)  
*Metric constructor.*
- [Metric](#) ([Metric](#) const &metric)  
*Metric copy constructor.*
- [~Metric](#) ()  
*Metric default destructor.*
- string [getName](#) () const  
*Metric name accessor.*
- string [getType](#) () const

*Metric* type accessor.

- string `getUnit ()` const

*Metric* unit accessor.

- unsigned long int `getIntegerValue ()` const

*Metric* integer value accessor.

- long double `getFloatValue ()` const

*Metric* real value accessor.

- void `nameToLower ()`

*Metric* object name property character case conversion function to lower case.

- long double `multiplier ()`

- void `setName (const string &name)`

*Metric* name setting method.

- void `setType (const string &rtype)`

*Metric* type setting method.

- void `setUnit (const string &runit)`

*Metric* unit setting method.

- void `setValue (const unsigned long int &rvalue)`

*Metric* integer value setting method.

- void `setValue (const long double &rvalue)`

*Metric* floating point (real) value setting method.

- `Metric & operator= (const Metric &m)`

*Metric* assignment operator.

## Friends

- ostream & `operator<<` (ostream &os, const `Metric` &m)

XML output stream operator (*Metric* object XML serializer).

- ostream & `operator<<` (ostream &os, const `Metric` \*m)

XML output stream operator (*Metric* object XML serializer).

## Data Structures

- union `Value`



### 6.7.1 Detailed Description

[Metric](#) Class which captures the design metric concept.

**Author:**

Gerardo de Miguel González

**Version:**

1.0

**Date:**

June 2008

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 [Metric::Metric](#) ()

[Metric](#) default constructor.

**Returns:**

A [Metric](#) Object

#### 6.7.2.2 [Metric::Metric](#) (const string & *rname*, const string & *rtype*, const string & *runit*)

[Metric](#) constructor.

**Parameters:**

*rname* metric name (p.e execution\_time)

*rtype* metric type (p.e float)

*runit* metric unit (p.e seconds)

**Returns:**

A [Metric](#) object

#### 6.7.2.3 [Metric::Metric](#) ([Metric](#) const & *m*)

[Metric](#) copy constructor.

**Parameters:**

*m* a reference to a [Metric](#) object

**Returns:**

A [Metric](#) object

#### 6.7.2.4 [Metric::~~Metric](#) ()

[Metric](#) default destructor.

### 6.7.3 Member Function Documentation

#### 6.7.3.1 string Metric::getName () const

[Metric](#) name accessor.

**Returns:**

A string object which holds the name property content of the [Metric](#) object

#### 6.7.3.2 string Metric::getType () const

[Metric](#) type accessor.

**Returns:**

A string object which holds the type property content of the [Metric](#) object

#### 6.7.3.3 string Metric::getUnit () const

[Metric](#) unit accessor.

**Returns:**

A string object which holds the unit property content of the [Metric](#) object

#### 6.7.3.4 unsigned long int Metric::getIntegerValue () const

[Metric](#) integer value accessor.

**Returns:**

An unsigned integer number which holds the integer value of the [Metric](#) object

#### 6.7.3.5 long double Metric::getFloatValue () const

[Metric](#) real value accessor.

**Returns:**

A floating point number which holds the real value of the [Metric](#) object

#### 6.7.3.6 void Metric::nameToLower ()

[Metric](#) object name property character case conversion function to lower case.

#### 6.7.3.7 long double Metric::multiplier ()

**6.7.3.8 void Metric::setName (const string & *rname*)**

[Metric](#) name setting method.

**Parameters:**

*rname* a constant reference to a string object holding a name to set

**6.7.3.9 void Metric::setType (const string & *rtype*)**

[Metric](#) type setting method.

**Parameters:**

*rtype* a constant reference to a string object holding a type to set

**6.7.3.10 void Metric::setUnit (const string & *runit*)**

[Metric](#) unit setting method.

**Parameters:**

*runit* a constant reference to a string object holding a unit to set

**6.7.3.11 void Metric::setValue (const unsigned long int & *rvalue*)**

[Metric](#) integer value setting method.

**Parameters:**

*rvalue* a constant reference to an integer number holding a value to be set

**6.7.3.12 void Metric::setValue (const long double & *rvalue*)**

[Metric](#) floating point (real) value setting method.

**Parameters:**

*rvalue* a constant reference to floating-point number holding a value to be set

**6.7.3.13 Metric & Metric::operator= (const Metric & *m*)**

[Metric](#) assignment operator.

**Parameters:**

*m* a constant reference to a [Metric](#) object (rvalue)

**Returns:**

a reference to a [Metric](#) object (lvalue)

### 6.7.4 Friends And Related Function Documentation

#### 6.7.4.1 ostream& operator<< (ostream & *os*, const Metric & *m*) [friend]

XML output stream operator ([Metric](#) object XML serializer).

**Parameters:**

- os* an output stream object reference
- m* a constant [Metric](#) object reference

**Returns:**

a reference to an output stream object

#### 6.7.4.2 ostream& operator<< (ostream & *os*, const Metric \* *m*) [friend]

XML output stream operator ([Metric](#) object XML serializer).

**Parameters:**

- os* an output stream object reference
- m* a constant [Metric](#) object pointer

**Returns:**

a reference to an output stream object

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

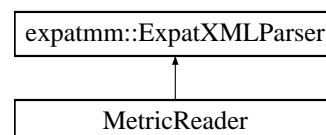
- [metric.h](#)
- [metric.cpp](#)

## 6.8 MetricReader Class Reference

[Metric](#) (expat + [expatmm](#)) Parser Class.

```
#include <parse.h>
```

Inheritance diagram for MetricReader::



### Public Member Functions

- [MetricReader](#) ()  
*MetricReader* default constructor.

- **MetricReader** (const char \*xmlFile)  
*MetricReader constructor (from a XML file source).*
- **~MetricReader** ()  
*Metric default destructor.*
- bool **writeXMLFile** (const char \*xmlOutputFile)  
*writes an XML file with the Metric objects as specified in MULTICUBE*
- bool **writeXMLErrorFile** (const char \*xmlOutputFile, const char \*reason, const char \*kind)  
*writes an XML error file as specified in MULTICUBE*
- **Metric** **getMetric** ()  
*Metric object accessor Destructive retrieval of a Metric object from the Stack.*
- void **setMetric** (Metric &m)  
*Metric object setting method Insertion of a Metric object on the Stack.*
- bool **isMetricLeft** () const  
*checks whether there is some Metric object left in the Stack*
- int **size** () const  
*size of the Stack holding the Metric objects*

### Protected Member Functions

- virtual ssize\_t **read\_block** (void)  
*expatmm (expat C++ wrapper) read\_block method implementation Read the XML source (i.e a file) getting blocks of characters from a predefined block size which are then parsed and reported as events (SAX parsing style) which are handled with the start and end tag event handlers*
- virtual void **StartElement** (const XML\_Char \*name, const XML\_Char \*\*attrs)  
*StartElement Handler (SAX Parsing style) which triggers after parsing a starting tag in the XML file The data got from the attribute/value pairs is used to build a Metric object and the Metric object is pushed into a Stack (queued) container.*
- virtual void **EndElement** (const XML\_Char \*name)  
*EndElement Handler (SAX Parsing style) which triggers after parsing an ending tag in the XML file.*

#### 6.8.1 Detailed Description

**Metric** (expat + expatmm) Parser Class.

#### Author:

Gerardo de Miguel González

#### Version:

1.0

**Date:**

June 2008

**6.8.2 Constructor & Destructor Documentation****6.8.2.1 MetricReader::MetricReader ()**

[MetricReader](#) default constructor.

**Returns:**

A [MetricReader](#) Object

**Warning:**

The default constructor is disabled declaring it private

**6.8.2.2 MetricReader::MetricReader (const char \* *xmlFile*)**

[MetricReader](#) constructor (from a XML file source).

**Parameters:**

*xmlfile* XML file name where the [MetricReader](#) object is built from

**Returns:**

A [MetricReader](#) object

**6.8.2.3 MetricReader::~~MetricReader ()**

[Metric](#) default destructor.

**6.8.3 Member Function Documentation****6.8.3.1 bool MetricReader::writeXMLFile (const char \* *xmlOutputFile*)**

writes an XML file with the [Metric](#) objects as specified in MULTICUBE

**Parameters:**

*xmlOutputFile* constant char array holding the name of the XML output file

**Returns:**

A boolean result 'True' if the writing process has been successful

**6.8.3.2 bool MetricReader::writeXMLErrorFile (const char \* *xmlOutputFile*, const char \* *reason*, const char \* *kind*)**

writes an XML error file as specified in MULTICUBE

**Parameters:**

*xmlOutputFile* constant char array holding the name of the XML output file

*reason* a word description of the exception or error reported

*kind* the severity grade of the error reported (i.e fatal)

**Returns:**

A boolean result 'True' if the writing process has been successful

**6.8.3.3 Metric MetricReader::getMetric ()**

[Metric](#) object accessor Destructive retrieval of a [Metric](#) object from the Stack.

**Returns:**

A [Metric](#) object

**6.8.3.4 void MetricReader::setMetric (Metric & m)**

[Metric](#) object setting method Insertion of a [Metric](#) object on the Stack.

**Parameters:**

*m* the [Metric](#) object that is going to be pushed into the Stack

**6.8.3.5 bool MetricReader::isMetricLeft () const**

checks whether there is some [Metric](#) object left in the Stack

**Returns:**

A boolean result 'True' if there is some [Metric](#) object left

**6.8.3.6 int MetricReader::size () const**

size of the Stack holding the [Metric](#) objects

**Returns:**

the number of [Metric](#) objects which are in the Stack

**6.8.3.7 ssize\_t MetricReader::read\_block (void) [protected, virtual]**

[expatmm](#) (expat C++ wrapper) read\_block method implementation Read the XML source (i.e a file) getting blocks of characters from a predefined block size which are then parsed and reported as events (SAX parsing style) which are handled with the start and end tag event handlers

**Returns:**

the number of blocks read or '-1' if there is some error

Reimplemented from [expatmm::ExpatXMLParser](#).

**6.8.3.8 void MetricReader::StartElement (const XML\_Char \* *name*, const XML\_Char \*\* *attrs*)**  
[protected, virtual]

StartElement Handler (SAX Parsing style) which triggers after parsing a starting tag in the XML file The data got from the attribute/value pairs is used to build a [Metric](#) object and the [Metric](#) object is pushed into a Stack (queued) container.

**Parameters:**

*name* constant char array which holds the tag's name which is parsed

*attrs* constant Nx2 array which holds the attribute/value pairs within the XML tag which is parsed

**Returns:**

Reimplemented from [expatmm::ExpatXMLParser](#).

**6.8.3.9 void MetricReader::EndElement (const XML\_Char \* *name*)** [protected, virtual]

EndElement Handler (SAX Parsing style) which triggers after parsing an ending tag in the XML file.

**Parameters:**

*name* constant char array which holds the tag's name which is parsed

**Returns:**

Reimplemented from [expatmm::ExpatXMLParser](#).

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

- [parse.h](#)
- [parse.cpp](#)

## 6.9 sw\_allocation Struct Reference

```
#include <xml_if.h>
```

**Data Fields**

- char \* [name](#)
- char \* [path](#)
- char \* [priority](#)
- char \* [policy](#)
- char \* [args](#)
- char \* [task\\_name](#)
- char \* [os\\_name](#)
- char \* [resource\\_name](#)
- struct [sw\\_task](#) \* [task](#)



- struct [hw\\_instance](#) \* [hw\\_resource](#)
- struct [hw\\_group](#) \* [hw\\_group](#)
- struct [sw\\_instance](#) \* [sw\\_resource](#)
- int [offset](#)
- void \* [scope\\_data](#)

### 6.9.1 Field Documentation

**6.9.1.1** char\* [sw\\_allocation::name](#)

**6.9.1.2** char\* [sw\\_allocation::path](#)

**6.9.1.3** char\* [sw\\_allocation::priority](#)

**6.9.1.4** char\* [sw\\_allocation::policy](#)

**6.9.1.5** char\* [sw\\_allocation::args](#)

**6.9.1.6** char\* [sw\\_allocation::task\\_name](#)

**6.9.1.7** char\* [sw\\_allocation::os\\_name](#)

**6.9.1.8** char\* [sw\\_allocation::resource\\_name](#)

**6.9.1.9** struct [sw\\_task](#)\* [sw\\_allocation::task](#) [read]

**6.9.1.10** struct [hw\\_instance](#)\* [sw\\_allocation::hw\\_resource](#) [read]

**6.9.1.11** struct [hw\\_group](#)\* [sw\\_allocation::hw\\_group](#) [read]

**6.9.1.12** struct [sw\\_instance](#)\* [sw\\_allocation::sw\\_resource](#) [read]

**6.9.1.13** int [sw\\_allocation::offset](#)

**6.9.1.14** void\* [sw\\_allocation::scope\\_data](#)

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

- [xml\\_if.h](#)

## 6.10 sw\_component Struct Reference

```
#include <xml_if.h>
```

## Data Fields

- char \* [name](#)
- char \* [path](#)
- char \* [type](#)

### 6.10.1 Field Documentation

#### 6.10.1.1 char\* sw\_component::name

#### 6.10.1.2 char\* sw\_component::path

#### 6.10.1.3 char\* sw\_component::type

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

- [xml\\_if.h](#)

## 6.11 sw\_instance Struct Reference

```
#include <xml_if.h>
```

## Data Fields

- char \* [name](#)
- char \* [path](#)
- char \* [type](#)
- char \* [component\\_name](#)
- char \* [resource\\_name](#)
- struct [sw\\_component](#) \* [component](#)
- struct [hw\\_instance](#) \* [hw\\_resource](#)
- struct [hw\\_group](#) \* [hw\\_group](#)
- int [offset](#)
- void \* [scope\\_data](#)

### 6.11.1 Field Documentation

#### 6.11.1.1 char\* sw\_instance::name

#### 6.11.1.2 char\* sw\_instance::path

#### 6.11.1.3 char\* sw\_instance::type

#### 6.11.1.4 char\* sw\_instance::component\_name

#### 6.11.1.5 char\* sw\_instance::resource\_name

**6.11.1.6** struct sw\_component\* sw\_instance::component [read]

**6.11.1.7** struct hw\_instance\* sw\_instance::hw\_resource [read]

**6.11.1.8** struct hw\_group\* sw\_instance::hw\_group [read]

**6.11.1.9** int sw\_instance::offset

**6.11.1.10** void\* sw\_instance::scope\_data

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

- [xml\\_if.h](#)

## 6.12 sw\_task Struct Reference

```
#include <xml_if.h>
```

### Data Fields

- char \* [name](#)
- char \* [path](#)
- char \* [type](#)
- char \* [function](#)
- char \* [file](#)
- char \* [class\\_name](#)
- char \* [compute\\_time](#)
- char \* [period](#)
- char \* [data\\_size](#)
- char \* [init\\_time](#)
- char \* [fin\\_time](#)

### 6.12.1 Field Documentation

**6.12.1.1** char\* sw\_task::name

**6.12.1.2** char\* sw\_task::path

**6.12.1.3** char\* sw\_task::type

**6.12.1.4** char\* sw\_task::function

**6.12.1.5** char\* sw\_task::file

**6.12.1.6** char\* sw\_task::class\_name

**6.12.1.7** char\* sw\_task::compute\_time

**6.12.1.8** char\* sw\_task::period

**6.12.1.9** char\* sw\_task::data\_size

**6.12.1.10** char\* sw\_task::init\_time

**6.12.1.11** char\* sw\_task::fin\_time

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

- [xml\\_if.h](#)

## 6.13 xml\_basic\_info Struct Reference

```
#include <xml_hierarchy.h>
```

### Data Fields

- char \* [name](#)
- char \*\* [args](#)
- struct [xml\\_basic\\_info](#) \* [parent](#)
- struct [list\\_elem](#) \* [children\\_header](#)
- char [index](#)
- char \* [init\\_index](#)
- char \* [repeat](#)

### 6.13.1 Field Documentation

**6.13.1.1** char\* xml\_basic\_info::name

**6.13.1.2** char\*\* xml\_basic\_info::args

**6.13.1.3** struct xml\_basic\_info\* xml\_basic\_info::parent [read]

**6.13.1.4** struct list\_elem\* xml\_basic\_info::children\_header [read]

**6.13.1.5** char xml\_basic\_info::index

**6.13.1.6** char\* xml\_basic\_info::init\_index

#### 6.13.1.7 char\* xml\_basic\_info::repeat

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

- [xml\\_hierarchy.h](#)

## 6.14 xml\_parameter\_info Struct Reference

```
#include <xml_configuration_file.h>
```

### Data Fields

- char \* [name](#)
- char \* [value](#)

#### 6.14.1 Field Documentation

##### 6.14.1.1 char\* xml\_parameter\_info::name

##### 6.14.1.2 char\* xml\_parameter\_info::value

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

- [xml\\_configuration\\_file.h](#)

## 6.15 xml\_scope\_simulation\_info\_t Struct Reference

```
#include <uc_load_xml.h>
```

### Data Fields

- long long [simulation\\_time](#)
- enum sc\_time\_unit [sim\\_time\\_unit](#)
- int [debug\\_level](#)
- int [warnings](#)

#### 6.15.1 Field Documentation

##### 6.15.1.1 long long xml\_scope\_simulation\_info\_t::simulation\_time

##### 6.15.1.2 enum sc\_time\_unit xml\_scope\_simulation\_info\_t::sim\_time\_unit

##### 6.15.1.3 int xml\_scope\_simulation\_info\_t::debug\_level

##### 6.15.1.4 int xml\_scope\_simulation\_info\_t::warnings

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

- [uc\\_load\\_xml.h](#)

## 7 File Documentation

### 7.1 expatmm-libdef.h File Reference

### 7.2 ExpatMM-version.cpp File Reference

```
#include <cstdio>
#include <string>
#include "expat.h"
#include "expatmm.h"
```

#### Functions

- std::string [expatmm::getExpatMMVersion](#) (void)

### 7.3 expatmm.h File Reference

```
#include "expatmm-libdef.h"
#include "ExpatXMLParser.h"
```

#### Namespaces

- namespace [expatmm](#)

#### Defines

- #define [EXPATMM\\_LIBRARY\\_MAJOR](#) 1
- #define [EXPATMM\\_LIBRARY\\_MINOR](#) 0
- #define [EXPATMM\\_LIBRARY\\_REVISION](#) 0
- #define [EXPATMM\\_LIBRARY\\_VERSION](#) "1.0.0"

#### Functions

- std::string [expatmm::getExpatMMVersion](#) (void)

#### 7.3.1 Define Documentation

**7.3.1.1** #define [EXPATMM\\_LIBRARY\\_MAJOR](#) 1

**7.3.1.2** #define [EXPATMM\\_LIBRARY\\_MINOR](#) 0

**7.3.1.3** #define [EXPATMM\\_LIBRARY\\_REVISION](#) 0

**7.3.1.4** #define [EXPATMM\\_LIBRARY\\_VERSION](#) "1.0.0"

## 7.4 ExpatXMLParser.cpp File Reference

```
#include <cstdint>
#include <cstring>
#include <iostream>
#include "expat.h"
#include "expat_external.h"
#include "ExpatXMLParser.h"
```

## 7.5 ExpatXMLParser.h File Reference

```
#include "expatmm-libdef.h"
```

### Namespaces

- namespace [expatmm](#)

### Data Structures

- class [expatmm::ExpatXMLParser](#)

### Defines

- #define [XML\\_CHUNK\\_SIZE](#) 10240

#### 7.5.1 Define Documentation

##### 7.5.1.1 #define XML\_CHUNK\_SIZE 10240

## 7.6 ExpatXMLParser.h File Reference

```
#include "expatmm-libdef.h"
```

### Namespaces

- namespace [expatmm](#)

### Data Structures

- class [expatmm::ExpatXMLParser](#)

### Defines

- #define [XML\\_CHUNK\\_SIZE](#) 10240

### 7.6.1 Define Documentation

#### 7.6.1.1 #define XML\_CHUNK\_SIZE 10240

## 7.7 metric.cpp File Reference

```
#include "metric.h"
```

### Functions

- ostream & [operator<<](#) (ostream &os, const [Metric](#) &m)  
*XML output stream operator ([Metric](#) object XML serializer).*
- ostream & [operator<<](#) (ostream &os, const [Metric](#) \*m)  
*XML output stream operator ([Metric](#) object XML serializer).*

### 7.7.1 Function Documentation

#### 7.7.1.1 ostream& operator<< (ostream & os, const Metric \* m)

XML output stream operator ([Metric](#) object XML serializer).

##### Parameters:

- os* an output stream object reference  
*m* a constant [Metric](#) object pointer

##### Returns:

a reference to an output stream object

#### 7.7.1.2 ostream& operator<< (ostream & os, const Metric & m)

XML output stream operator ([Metric](#) object XML serializer).

##### Parameters:

- os* an output stream object reference  
*m* a constant [Metric](#) object reference

##### Returns:

a reference to an output stream object

## 7.8 metric.h File Reference

```
#include <iostream>  
#include <string>  
#include <iomanip>
```



### Data Structures

- class [Metric](#)  
*Metric Class which captures the design metric concept.*
- union **Metric::Value**

## 7.9 parse.cpp File Reference

```
#include <cstddef>
#include <cstring>
#include <sys/types.h>
#include <strings.h>
#include "metric.h"
#include "expat.h"
#include "expatmm.h"
#include "parse.h"
```

## 7.10 parse.h File Reference

```
#include <fstream>
#include <queue>
#include "ExpatXMLParser.h"
```

### Data Structures

- class [MetricReader](#)  
*Metric (expat + expatmm) Parser Class.*

## 7.11 uc\_create\_xml\_platform.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
#include <ctype.h>
#include <vector>
#include "uc_load_xml.h"
#include "uc_create_xml_platform.h"
#include "xml_input.h"
#include "xml_if.h"
#include "xml_list.h"
```

```
#include "sc_scope.h"
#include "uc_unistd.h"
#include "uc_hal_sw.h"
```

## Functions

- void \* [get\\_dynamic\\_function](#) (char \*file, char \*function)  
*Obtain the pointer to a function from a dynamic library, knowing the function name and the file name. The function has to be 'extern "C"'.*
- void \* [get\\_static\\_function](#) (char \*function)  
*Obtain the pointer to a function from a the executable itself, knowing the function name and the file name. Tries with the name in C and C++ for gcc 3.x-4.x.*
- void [load\\_task\\_info](#) (struct \_xml\_intermediate\_info \*info, struct [sw\\_allocation](#) \*alloc)  
*Load the information required to execute a function from the [sw\\_allocation](#) struct, where all are strings, to the \_xml\_intermediate\_info, as function pointer, priority, policy, arguments, ...*
- void [load\\_taskload\\_info](#) (struct \_xml\_taskload\_info \*info, struct [sw\\_allocation](#) \*alloc)  
*Create the data structure to create a task load from the xml structure containing xml strings.*
- void \* [\\_xml\\_intermediate\\_function](#) (void \*data)  
*Intermediate function used to launch a SW task. Read the data struct and starts the function indicated in the function pointer.*
- void \* [\\_xml\\_taskload\\_function](#) (void \*data)  
*Function executed to simulate a task load.*
- void [instanciate\\_sw\\_components](#) ()  
*Instantiate the SW components (OS) described in the XML System Description file.*
- void [instanciate\\_hw\\_components](#) ()  
*Instantiate the HW components described in the XML System Description file: processors, buses, memories, ...*
- void [connect\\_hw\\_components](#) ()  
*Connect the HW components of the system as described in the XML System Description file.*
- void [allocate\\_sw\\_tasks](#) ()  
*Function executed to instanciate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. If it is a driver, it is adequately load.*

## Variables

- char \* [executable\\_name](#)
- long long [uc\\_segment\\_time](#)
- std::vector< UC\_TLM\_bus\_class \* > [system\\_buses](#)

### 7.11.1 Function Documentation

#### 7.11.1.1 void\* \_xml\_intermediate\_function (void \* *data*)

Intermediate function used to launch a SW task. Read the data struct and starts the function indicated in the function pointer.

**Parameters:**

*data* Structure where the information is stored

**Returns:**

Return value of the called function

#### 7.11.1.2 void\* \_xml\_taskload\_function (void \* *data*)

Function executed to simulate a task load.

**Parameters:**

*data* Structure where the taskload information is stored

**Returns:**

NULL, used for compatibility

#### 7.11.1.3 void allocate\_sw\_tasks ()

Function executed to instantiate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. If it is a driver, it is adequately loaded.

**Parameters:**

*\return*

#### 7.11.1.4 void connect\_hw\_components ()

Connect the HW components of the system as described in the XML System Description file.

Current version only connections to connect the network interface to the network

#### 7.11.1.5 void\* get\_dynamic\_function (char \* *file*, char \* *function*)

Obtain the pointer to a function from a dynamic library, knowing the function name and the file name. The function has to be 'extern "C"'.

**Parameters:**

*file* Name of the dynamic library where the function is

*function* Name of the function

**Returns:**

The function pointer

**7.11.1.6 void\* get\_static\_function (char \*function)**

Obtain the pointer to a function from a the executable itself, knowing the function name and the file name. Tries with the name in C and C++ for gcc 3.x-4.x.

**Parameters:**

*function* Name of the function

**Returns:**

The function pointer

**7.11.1.7 void instantiate\_hw\_components ()**

Instantiate the HW components described in the XML System Description file: processors, buses, memories, ...

**Returns:****7.11.1.8 void instantiate\_sw\_components ()**

Instantiate the SW components (OS) described in the XML System Description file.

**Returns:****7.11.1.9 void load\_task\_info (struct \_xml\_intermediate\_info \*info, struct sw\_allocation \*alloc)**

Load the information required to execute a function from the [sw\\_allocation](#) struct, where all are strings, to the \_xml\_intermediate\_info, as function pointer, priority, policy, arguments, ...

**Parameters:**

*info* Structure where the information is stored

*alloc* Structure from where the information is read

**Returns:****7.11.1.10 void load\_taskload\_info (struct \_xml\_taskload\_info \*info, struct sw\_allocation \*alloc)**

Create the data structure to create a task load from the xml structure containing xml strings.

**Parameters:**

*info* Structure where the information is stored

*alloc* Structure from where the information is read

**Returns:**

### 7.11.2 Variable Documentation

#### 7.11.2.1 char\* executable\_name

#### 7.11.2.2 std::vector<UC\_TLM\_bus\_class\*> system\_buses

#### 7.11.2.3 long long uc\_segment\_time

## 7.12 uc\_create\_xml\_platform.h File Reference

### Functions

- void [instanciate\\_sw\\_components](#) ()  
*Instantiate the SW components (OS) described in the XML System Description file.*
- void [instanciate\\_hw\\_components](#) ()  
*Instantiate the HW components described in the XML System Description file: processors, buses, memories, ...*
- void [connect\\_hw\\_components](#) ()  
*Connect the HW components of the system as described in the XML System Description file.*
- void [allocate\\_sw\\_tasks](#) ()  
*Function executed to instantiate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. If it is a driver, it is adequately load.*

### 7.12.1 Function Documentation

#### 7.12.1.1 void allocate\_sw\_tasks ()

Function executed to instantiate a SW task. If the task is a SW task or a taskload, a process is created in the corresponding thread. If it is a driver, it is adequately load.

#### Parameters:

*\return*

#### 7.12.1.2 void connect\_hw\_components ()

Connect the HW components of the system as described in the XML System Description file.

Current version only connections to connect the network interface to the network

#### 7.12.1.3 void instanciate\_hw\_components ()

Instantiate the HW components described in the XML System Description file: processors, buses, memories, ...

#### Returns:

#### 7.12.1.4 void instantiate\_sw\_components ()

Instantiate the SW components (OS) described in the XML System Description file.

**Returns:**

### 7.13 uc\_load\_xml.cpp File Reference

```
#include "sc_scope.h"
#include "xml_main.h"
#include "uc_load_xml.h"
#include "uc_create_xml_platform.h"
```

#### Functions

- int [sc\\_main](#) (int argc, char \*\*argv)  
*SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.*
- void [destroy\\_objects](#) ()  
*Destroy the dynamically allocated simulation objects and print out the results on screen.*
- void [usage](#) (char \*programa)  
*Prints out the program command-line help.*
- void [load\\_xml\\_platform\\_file](#) (char \*file)  
*Execute all the functions required to read the XML System Description file and create the system model.*
- void [load\\_xml\\_configuration\\_file](#) (char \*xml\_file)  
*Load the XML System Configuration file.*

#### Variables

- UC\_gui\_connector \* [gui\\_connector](#)
- [xml\\_scope\\_simulation\\_info\\_t](#) [xml\\_scope\\_simulation\\_info](#) = {-1, SC\_NS, 0}
- char \* [executable\\_name](#)
- int [activate\\_backtrace](#)
- vector< UC\_rtos\_class \* > [rtos\\_list](#)
- vector< UC\_NoC\_Interface \* > [simulator\\_list](#)

#### 7.13.1 Function Documentation

##### 7.13.1.1 void destroy\_objects ()

Destroy the dynamically allocated simulation objects and print out the results on screen.

**7.13.1.2 void load\_xml\_configuration\_file (char \* *xml\_file*)**

Load the XML System Configuration file.

**Parameters:**

*file* Name of the XML System Configuration file

**7.13.1.3 void load\_xml\_platform\_file (char \* *file*)**

Execute all the functions required to read the XML System Description file and create the system model.

**Parameters:**

*file* Name of the XML System Description file

**7.13.1.4 int sc\_main (int *argc*, char \*\* *argv*)**

SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.

**Parameters:**

*argc* Number of arguments received from the terminal

*argv* Arguments received from the terminal

**Returns:**

Program result

**7.13.1.5 void usage (char \* *programa*)**

Prints out the program command-line help.

**7.13.2 Variable Documentation****7.13.2.1 int activate\_backtrace****7.13.2.2 char\* executable\_name****7.13.2.3 UC\_gui\_connector\* gui\_connector****7.13.2.4 vector<UC\_rtos\_class \*> rtos\_list****7.13.2.5 vector<UC\_NoC\_Interface \*> simulator\_list****7.13.2.6 xml\_scope\_simulation\_info\_t xml\_scope\_simulation\_info = {-1, SC\_NS, 0}**

## 7.14 uc\_load\_xml.h File Reference

```
#include "sc_scope.h"
#include <vector>
#include "expat.h"
#include "expatmm.h"
#include "metric.h"
#include "parse.h"
```

### Data Structures

- struct [xml\\_scope\\_simulation\\_info\\_t](#)

### Functions

- int [sc\\_main](#) (int argc, char \*\*argv)  
*SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.*
- void [usage](#) (char \*program)  
*Prints out the program command-line help.*
- void [load\\_xml\\_platform\\_file](#) (char \*xml\_config)  
*Execute all the functions required to read the XML System Description file and create the system model.*
- void [load\\_xml\\_configuration\\_file](#) (char \*xml\_file)  
*Load the XML System Configuration file.*
- int [obtain\\_xml\\_metrics](#) ([MetricReader](#) &mr)
- void [destroy\\_objects](#) ()  
*Destroy the dynamically allocated simulation objects and print out the results on screen.*

### Variables

- [xml\\_scope\\_simulation\\_info\\_t](#) [xml\\_scope\\_simulation\\_info](#)
- vector< UC\_rtos\_class \* > [rtos\\_list](#)
- vector< UC\_NoC\_Interface \* > [simulator\\_list](#)
- UC\_gui\_connector \* [gui\\_connector](#)

#### 7.14.1 Function Documentation

##### 7.14.1.1 void destroy\_objects ()

Destroy the dynamically allocated simulation objects and print out the results on screen.



**7.14.1.2 void load\_xml\_configuration\_file (char \* *xml\_file*)**

Load the XML System Configuration file.

**Parameters:**

*file* Name of the XML System Configuration file

**7.14.1.3 void load\_xml\_platform\_file (char \* *file*)**

Execute all the functions required to read the XML System Description file and create the system model.

**Parameters:**

*file* Name of the XML System Description file

**7.14.1.4 int obtain\_xml\_metrics (MetricReader & *mr*)****7.14.1.5 int sc\_main (int *argc*, char \*\* *argv*)**

SystemC main function. It decodes the input arguments, reads the xml files, builds the system model, and starts the simulation.

**Parameters:**

*argc* Number of arguments received from the terminal

*argv* Arguments received from the terminal

**Returns:**

Program result

**7.14.1.6 void usage (char \* *program*)**

Prints out the program command-line help.

**7.14.2 Variable Documentation****7.14.2.1 UC\_gui\_connector\* *gui\_connector*****7.14.2.2 vector<UC\_rtos\_class \*> *rtos\_list*****7.14.2.3 vector<UC\_NoC\_Interface \*> *simulator\_list*****7.14.2.4 xml\_scope\_simulation\_info\_t *xml\_scope\_simulation\_info***

## 7.15 xml\_configuration\_file.c File Reference

```
#include "xml_configuration_file.h"
#include "xml_list.h"
#include "xml_if.h"
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
```

### Functions

- const char \* [xml\\_load\\_configuration\\_parameter](#) (const char \*param)  
*Get the corresponding value for a configuration parameter.*
- void [xml\\_add\\_parameter](#) (char \*userData, const char \*name, const char \*\*atts)  
*Decode the attributes of a configuration parameter xml clause.*
- void [xml\\_end\\_parameter](#) (char \*userData, const char \*name)  
*Close a configuration xml clause.*

### Variables

- struct [list\\_elem](#) \* [xml\\_config\\_parameter\\_header](#) = 0
- struct [xml\\_parameter\\_info](#) \* [prev\\_xml\\_parameter\\_info](#) = NULL

### 7.15.1 Function Documentation

#### 7.15.1.1 void [xml\\_add\\_parameter](#) (char \* *userData*, const char \* *name*, const char \*\* *atts*)

Decode the attributes of a configuration parameter xml clause.

##### Parameters:

*userData* Pointer to the user defined buffer  
*name* Name of the parameter xml clause  
*atts* Attributes of the xml clause

##### Returns:

#### 7.15.1.2 void [xml\\_end\\_parameter](#) (char \* *userData*, const char \* *name*)

Close a configuration xml clause.

##### Parameters:

*userData* Pointer to the user defined buffer

*name* Name of the parameter xml clause

**Returns:**

#### 7.15.1.3 const char\* xml\_load\_configuration\_parameter (const char \* *param*)

Get the corresponding value for a configuration parameter.

**Parameters:**

*param* Name of the parameter

**Returns:**

Parameter value

### 7.15.2 Variable Documentation

#### 7.15.2.1 struct xml\_parameter\_info\* prev\_xml\_parameter\_info = NULL

#### 7.15.2.2 struct list\_elem\* xml\_config\_parameter\_header = 0

## 7.16 xml\_configuration\_file.h File Reference

### Data Structures

- struct [xml\\_parameter\\_info](#)

### Functions

- const char \* [xml\\_load\\_configuration\\_parameter](#) (const char \*param)  
*Get the corresponding value for a configuration parameter.*
- void [xml\\_add\\_parameter](#) (char \*userData, const char \*name, const char \*\*atts)  
*Decode the attributes of a configuration parameter xml clause.*
- void [xml\\_end\\_parameter](#) (char \*userData, const char \*name)  
*Close a configuration xml clause.*

### 7.16.1 Function Documentation

#### 7.16.1.1 void xml\_add\_parameter (char \* *userData*, const char \* *name*, const char \*\* *atts*)

Decode the attributes of a configuration parameter xml clause.

**Parameters:**

*userData* Pointer to the user defined buffer

*name* Name of the parameter xml clause

*atts* Attributes of the xml clause

**Returns:**

#### 7.16.1.2 void xml\_end\_parameter (char \* *userData*, const char \* *name*)

Close a configuration xml clause.

**Parameters:**

*userData* Pointer to the user defined buffer

*name* Name of the parameter xml clause

**Returns:**

#### 7.16.1.3 const char\* xml\_load\_configuration\_parameter (const char \* *param*)

Get the corresponding value for a configuration parameter.

**Parameters:**

*param* Name of the parameter

**Returns:**

Parameter value

## 7.17 xml\_hierarchy.c File Reference

```
#include "xml_hierarchy.h"
#include "xml_configuration_file.h"
#include "xml_list.h"
#include "xml_if.h"
#include "uc_load_xml.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
```

**Functions**

- void [xml\\_add\\_hierarchy](#) (char \**path*, const char \**name*, const char \*\**args*)  
*Add a new struct with the info of a clause to the correct point of the xml data hierarchy.*

- void `xml_up_hierarchy` (char \*path, const char \*name)  
*Close a clause moving the current hierarchy node to the upper node.*
- void `xml_increase_path` (char \*path, struct `xml_basic_info` \*elem, int index)  
*Increase the path when entering a new clause.*
- void `xml_decrease_path` (char \*path, const char \*name)  
*Decrease the path when exiting a new clause.*
- struct `xml_basic_info` \* `xml_search_path` (const char \*path)  
*Return the information struct of the clause indicated by the input path.*
- char \* `xml_check_name` (const char \*name, struct `xml_basic_info` \*prev\_elem, int index)  
*Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.*
- void `xml_display_hierarchy_node` (struct `xml_basic_info` \*info, char \*path, int index)  
*Create a node for a xml clause.*
- void `xml_display_hierarchy` (char \*path)  
*Create a node for a certain path.*
- const char \* `xml_string_index_convert` (const char \*data\_in)  
*Check if the argument is a configuratio parameter (start with '\_' ) or contain a "repeat" index and fix the string if required.*
- int `xml_chech_instance` (int index, const char \*\*atts)  
*Check if the clause has been selected in the implementation clause or must be discarded.*
- void `xml_select_implementation` (const char \*\*atts)  
*Decode the attributes of an implementation clause.*
- void `xml_simulation_parameters` (const char \*\*atts)  
*Decode the attributes of a simulation clause.*

## Variables

- struct `xml_basic_info` \* `xml_hierarchy_header` = 0
- struct `xml_basic_info` \* `xml_hierarchy_current` = 0
- int `xml_indexes` [30]
- char \* `xml_select_implementation_array` [11] = {NULL}

## 7.17.1 Function Documentation

### 7.17.1.1 void xml\_add\_hierarchy (char \*path, const char \*name, const char \*\*args)

Add a new struct with the info of a clause to the correct point of the xml data hierarchy.

#### Parameters:

*path* Clause path

*name* Clause name

*atts* Clause attributes

**Returns:**

#### 7.17.1.2 int xml\_check\_instance (int *index*, const char \*\* *atts*)

Check if the clause has been selected in the implementation clause or must be discarded.

**Parameters:**

*index* type of index

*atts* arguments of the clause

**Returns:**

1 if valid, 0 otherwise

#### 7.17.1.3 char\* xml\_check\_name (const char \* *name*, struct xml\_basic\_info \* *prev\_elem*, int *index*)

Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.

**Parameters:**

*name* Clause name

*prev\_elem* Clause info list

*index* used when creating several copies in a repeat clause

**Returns:**

The new clause name

#### 7.17.1.4 void xml\_decrease\_path (char \* *path*, const char \* *name*)

Decrease the path when exiting a new clause.

**Parameters:**

*path* Clause path

*elem* Clause info

**Returns:**

**7.17.1.5 void xml\_display\_hierarchy (char \* *path*)**

Create a node for a certain path.

**Parameters:**

*path* Clause path

**Returns:**

The new clause name

**7.17.1.6 void xml\_display\_hierarchy\_node (struct xml\_basic\_info \* *info*, char \* *path*, int *index*)**

Create a node for a xml clause.

**Parameters:**

*info* Clause info list

*name* Clause info

*index* used when creating several copies in a repeat clause

**Returns:**

The new clause name

**7.17.1.7 void xml\_increase\_path (char \* *path*, struct xml\_basic\_info \* *elem*, int *index*)**

Increase the path when entering a new clause.

**Parameters:**

*path* Clause path

*elem* Clause info

*index* used when creating several copies in a repeat clause

**Returns:****7.17.1.8 struct xml\_basic\_info\* xml\_search\_path (const char \* *path*)** [read]

Return the information struct of the clause indicated by the input path.

**Parameters:**

*path* Clause path

**Returns:**

The clause info

**7.17.1.9 void xml\_select\_implementation (const char \*\* *atts*)**

Decode the attributes of an implementation clause.

**Parameters:**

*atts* Clause attributes

**Returns:****7.17.1.10 void xml\_simulation\_parameters (const char \*\* *atts*)**

Decode the attributes of a simulation clause.

**Parameters:**

*atts* Clause attributes

**Returns:****7.17.1.11 const char\* xml\_string\_index\_convert (const char \* *data\_in*)**

Check if the argument is a configuratino parameter (start with '\_' ) or contain a "repeat" index and fix the string if required.

**Parameters:**

*data\_in* Input argument

**Returns:**

The fixed argument

**7.17.1.12 void xml\_up\_hierarchy (char \* *path*, const char \* *name*)**

Close a clause moving the current hierarchy node to the upper node.

**Parameters:**

*path* Clause path

*name* Clause name

**Returns:****7.17.2 Variable Documentation****7.17.2.1 struct xml\_basic\_info\* xml\_hierarchy\_current = 0**



7.17.2.2 struct xml\_basic\_info\* xml\_hierarchy\_header = 0

7.17.2.3 int xml\_indexes[30]

7.17.2.4 char\* xml\_select\_implementation\_array[11] = {NULL}

## 7.18 xml\_hierarchy.h File Reference

### Data Structures

- struct [xml\\_basic\\_info](#)

### Functions

- void [xml\\_display\\_hierarchy](#) (char \*path)  
*Create a node for a certain path.*
- void [xml\\_add\\_hierarchy](#) (char \*path, const char \*name, const char \*\*args)  
*Add a new struct with the info of a clause to the correct point of the xml data hierarchy.*
- void [xml\\_up\\_hierarchy](#) (char \*path, const char \*name)  
*Close a clause moving the current hierarchy node to the upper node.*
- void [xml\\_decrease\\_path](#) (char \*path, const char \*name)  
*Decrease the path when exiting a new clause.*
- void [xml\\_increase\\_path](#) (char \*path, struct [xml\\_basic\\_info](#) \*elem, int index)  
*Increase the path when entering a new clause.*
- struct [xml\\_basic\\_info](#) \* [xml\\_search\\_path](#) (const char \*path)  
*Return the information struct of the clause indicated by the input path.*
- char \* [xml\\_check\\_name](#) (const char \*name, struct [xml\\_basic\\_info](#) \*prev\_elem, int index)  
*Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.*
- const char \* [xml\\_string\\_index\\_convert](#) (const char \*data\_in)  
*Check if the argument is a configuratino parameter (start with '\_' ) or contain a "repeat" index and fix the string if required.*
- int [xml\\_chech\\_instance](#) (int i, const char \*\*atts)  
*Check if the clause has been selected in the implementation clause or must be discarded.*
- void [xml\\_select\\_implementation](#) (const char \*\*atts)  
*Decode the attributes of an implementation clause.*
- void [xml\\_simulation\\_parameters](#) (const char \*\*args)  
*Decode the attributes of a simulation clause.*

### 7.18.1 Function Documentation

#### 7.18.1.1 void xml\_add\_hierarchy (char \* *path*, const char \* *name*, const char \*\* *args*)

Add a new struct with the info of a clause to the correct point of the xml data hierarchy.

**Parameters:**

*path* Clause path  
*name* Clause name  
*atts* Clause attributes

**Returns:**

#### 7.18.1.2 int xml\_chech\_instance (int *index*, const char \*\* *atts*)

Check if the clause has been selected in the implementation clause or must be discarded.

**Parameters:**

*index* type of index  
*atts* arguments of the clause

**Returns:**

1 if valid, 0 otherwise

#### 7.18.1.3 char\* xml\_check\_name (const char \* *name*, struct xml\_basic\_info \* *prev\_elem*, int *index*)

Create a name, adding the corresponding suffix if more clauses with the same name hangs of the same node.

**Parameters:**

*name* Clause name  
*prev\_elem* Clause info list  
*index* used when creating several copies in a repeat clause

**Returns:**

The new clause name

#### 7.18.1.4 void xml\_decrease\_path (char \* *path*, const char \* *name*)

Decrease the path when exiting a new clause.

**Parameters:**

*path* Clause path  
*elem* Clause info

**Returns:**

**7.18.1.5 void xml\_display\_hierarchy (char \* *path*)**

Create a node for a certain path.

**Parameters:**

*path* Clause path

**Returns:**

The new clause name

**7.18.1.6 void xml\_increase\_path (char \* *path*, struct xml\_basic\_info \* *elem*, int *index*)**

Increase the path when entering a new clause.

**Parameters:**

*path* Clause path

*elem* Clause info

*index* used when creating several copies in a repeat clause

**Returns:****7.18.1.7 struct xml\_basic\_info\* xml\_search\_path (const char \* *path*)** [read]

Return the information struct of the clause indicated by the input path.

**Parameters:**

*path* Clause path

**Returns:**

The clause info

**7.18.1.8 void xml\_select\_implementation (const char \*\* *atts*)**

Decode the attributes of an implementation clause.

**Parameters:**

*atts* Clause attributes

**Returns:**

**7.18.1.9 void xml\_simulation\_parameters (const char \*\* *atts*)**

Decode the attributes of a simulation clause.

**Parameters:**

*atts* Clause attributes

**Returns:****7.18.1.10 const char\* xml\_string\_index\_convert (const char \* *data\_in*)**

Check if the argument is a configuratino parameter (start with '\_' ) or contain a "repeat" index and fix the string if required.

**Parameters:**

*data\_in* Input argument

**Returns:**

The fixed argument

**7.18.1.11 void xml\_up\_hierarchy (char \* *path*, const char \* *name*)**

Close a clause moving the current hierarchy node to the upper node.

**Parameters:**

*path* Clause path

*name* Clause name

**Returns:****7.19 xml\_if.c File Reference**

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "xml_if.h"
#include "xml_input.h"
#include "xml_list.h"
#include "xml_hierarchy.h"
```

## Functions

- int [strcasecmp\\_null](#) (const char \*str1, const char \*str2)  
*Compares two strings checking if any is NULL.*
- unsigned long int [atoi\\_null](#) (const char \*str)  
*Obtain a integer value from a string, checking if the string is null.*
- int [xml\\_declare\\_function](#) (char \*path, const char \*name, const char \*\*atts)  
*Check the type of clause, and call the corresponding function to decode the arguments.*
- void [xml\\_close\\_declare](#) (char \*path, const char \*name)  
*Close a xml clause.*
- void \* [xml\\_load\\_hw\\_component](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a HW\_Component clause.*
- void \* [xml\\_load\\_hw\\_instance](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a HW\_Instance clause.*
- void \* [xml\\_load\\_hw\\_connection](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a HW\_Connection clause.*
- void \* [xml\\_load\\_hw\\_group](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a Computing\_Group clause.*
- void \* [xml\\_add\\_to\\_hw\\_group](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a Computing\_Resource clause.*
- void \* [xml\\_load\\_sw\\_component](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a SW\_Component clause.*
- void \* [xml\\_load\\_sw\\_instance](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a SW instance clause.*
- void \* [xml\\_load\\_exec\\_component](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a Exec\_Component clause.*
- void \* [xml\\_load\\_exec\\_instance](#) (char \*path, const char \*\*attr)  
*Decode the arguments of a Exec\_Instance clause.*
- void [xml\\_add\\_process\\_argument](#) (char \*path, const char \*\*attr)  
*Add an argument to the previous sw allocation.*

### 7.19.1 Function Documentation

#### 7.19.1.1 unsigned long int [atoi\\_null](#) (const char \* str)

Obtain a integer value from a string, checking if the string is null.

**Parameters:**

*str* The string to be converted

**Returns:**

the value, or 0

**7.19.1.2 int strcasecmp\_null (const char \* *str1*, const char \* *str2*)**

Compares two strings checking if any is NULL.

**Parameters:**

*str1* String to be compared

*str2* String to be compared

**Returns:**

1 if equal, 0 otherwise.

**7.19.1.3 void xml\_add\_process\_argument (char \* *path*, const char \*\* *attr*)**

Add an argument to the previous sw allocation.

**Parameters:**

*path* Clause path

*name* Clause name

**7.19.1.4 void\* xml\_add\_to\_hw\_group (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a Computing\_Resource clause.

**Parameters:**

*path* Clause path

*name* Clause name

**Returns:**

the created struct

**7.19.1.5 void xml\_close\_declare (char \* *path*, const char \* *name*)**

Close a xml clause.

**Parameters:**

*path* Clause path

*name* Clause name

**Returns:**

**7.19.1.6 int xml\_declare\_function (char \* *path*, const char \* *name*, const char \*\* *atts*)**

Check the type of clause, and call the corresponding function to decode the arguments.

**Parameters:**

*path* Clause path  
*name* Clause name  
*atts* Clause attributes

**Returns:**

0 if ok, otherwise error

**7.19.1.7 void\* xml\_load\_exec\_component (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a Exec\_Component clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.19.1.8 void\* xml\_load\_exec\_instance (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a Exec\_Instance clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.19.1.9 void\* xml\_load\_hw\_component (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a HW\_Component clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.19.1.10 void\* xml\_load\_hw\_connection (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a HW\_Connection clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.19.1.11 void\* xml\_load\_hw\_group (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a Computing\_Group clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.19.1.12 void\* xml\_load\_hw\_instance (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a HW\_Instance clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.19.1.13 void\* xml\_load\_sw\_component (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a SW\_Component clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct



**7.19.1.14 void\* xml\_load\_sw\_instance (char \* path, const char \*\* attr)**

Decode the arguments of a SW instance clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20 xml\_if.h File Reference****Data Structures**

- struct [hw\\_component](#)
- struct [hw\\_instance](#)
- struct [hw\\_connection](#)
- struct [hw\\_group](#)
- struct [sw\\_component](#)
- struct [sw\\_instance](#)
- struct [sw\\_task](#)
- struct [sw\\_allocation](#)

**Functions**

- int [strcasecmp\\_null](#) (const char \*str1, const char \*str2)  
*Compares two strings checking if any is NULL.*
- unsigned long int [atoi\\_null](#) (const char \*str)  
*Obtain a integer value from a string, checking if the string is null.*
- void [xml\\_close\\_declare](#) (char \*path, const char \*name)  
*Close a xml clause.*
- int [xml\\_declare\\_function](#) (char \*path, const char \*name, const char \*\*atts)  
*Check the type of clause, and call the corresponding function to decode the arguments.*
- void \* [xml\\_load\\_hw\\_component](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a HW\_Component clause.*
- void \* [xml\\_load\\_hw\\_instance](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a HW\_Instance clause.*
- void \* [xml\\_load\\_hw\\_connection](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a HW\_Connection clause.*
- void \* [xml\\_load\\_hw\\_group](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a Computing\_Group clause.*

- void \* [xml\\_load\\_sw\\_component](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a SW\_Component clause.*
- void \* [xml\\_load\\_sw\\_instance](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a SW instance clause.*
- void \* [xml\\_load\\_exec\\_component](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a Exec\_Component clause.*
- void \* [xml\\_load\\_exec\\_instance](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a Exec\_Instance clause.*
- void \* [xml\\_add\\_to\\_hw\\_group](#) (char \*path, const char \*\*atts)  
*Decode the arguments of a Computing\_Resource clause.*
- void [xml\\_add\\_process\\_argument](#) (char \*path, const char \*\*attr)  
*Add an argument to the previous sw allocation.*
- void [xml\\_hw\\_component\\_string\\_convert](#) (struct [hw\\_component](#) \*component)
- void [xml\\_hw\\_instance\\_string\\_convert](#) (struct [hw\\_instance](#) \*instance)
- void [xml\\_hw\\_connection\\_string\\_convert](#) (struct [hw\\_connection](#) \*connection)
- void [xml\\_hw\\_group\\_string\\_convert](#) (struct [hw\\_group](#) \*group)
- void [xml\\_sw\\_instance\\_string\\_convert](#) (struct [sw\\_instance](#) \*instance)
- void [xml\\_sw\\_task\\_string\\_convert](#) (struct [sw\\_task](#) \*task)
- void [xml\\_sw\\_allocation\\_string\\_convert](#) (struct [sw\\_allocation](#) \*allocation)
- void [xml\\_connect\\_hw\\_instance](#) (struct [hw\\_instance](#) \*instance)
- void [xml\\_connect\\_hw\\_connection](#) (struct [hw\\_connection](#) \*connection)
- void [xml\\_connect\\_hw\\_group](#) (struct [hw\\_group](#) \*group)
- void [xml\\_connect\\_sw\\_instance](#) (struct [sw\\_instance](#) \*instance)
- int [xml\\_connect\\_sw\\_allocation](#) (struct [sw\\_allocation](#) \*allocation)

## Variables

- int [xml\\_info\\_level](#)

### 7.20.1 Function Documentation

#### 7.20.1.1 unsigned long int atoi\_null (const char \* str)

Obtain a integer value from a string, checking if the string is null.

#### Parameters:

*str* The string to be converted

#### Returns:

the value, or 0

**7.20.1.2 int strcmp\_null (const char \* *str1*, const char \* *str2*)**

Compares two strings checking if any is NULL.

**Parameters:**

*str1* String to be compared

*str2* String to be compared

**Returns:**

1 if equal, 0 otherwise.

**7.20.1.3 void xml\_add\_process\_argument (char \* *path*, const char \*\* *attr*)**

Add an argument to the previous sw allocation.

**Parameters:**

*path* Clause path

*name* Clause name

**7.20.1.4 void\* xml\_add\_to\_hw\_group (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a Computing\_Resource clause.

**Parameters:**

*path* Clause path

*name* Clause name

**Returns:**

the created struct

**7.20.1.5 void xml\_close\_declare (char \* *path*, const char \* *name*)**

Close a xml clause.

**Parameters:**

*path* Clause path

*name* Clause name

**Returns:****7.20.1.6 void xml\_connect\_hw\_connection (struct hw\_connection \* *connection*)****7.20.1.7 void xml\_connect\_hw\_group (struct hw\_group \* *group*)**

**7.20.1.8** void xml\_connect\_hw\_instance (struct hw\_instance \* *instance*)

**7.20.1.9** int xml\_connect\_sw\_allocation (struct sw\_allocation \* *allocation*)

**7.20.1.10** void xml\_connect\_sw\_instance (struct sw\_instance \* *instance*)

**7.20.1.11** int xml\_declare\_function (char \* *path*, const char \* *name*, const char \*\* *atts*)

Check the type of clause, and call the corresponding function to decode the arguments.

**Parameters:**

*path* Clause path  
*name* Clause name  
*atts* Clause attributes

**Returns:**

0 if ok, otherwise error

**7.20.1.12** void xml\_hw\_component\_string\_convert (struct hw\_component \* *component*)

**7.20.1.13** void xml\_hw\_connection\_string\_convert (struct hw\_connection \* *connection*)

**7.20.1.14** void xml\_hw\_group\_string\_convert (struct hw\_group \* *group*)

**7.20.1.15** void xml\_hw\_instance\_string\_convert (struct hw\_instance \* *instance*)

**7.20.1.16** void\* xml\_load\_exec\_component (char \* *path*, const char \*\* *attr*)

Decode the arguments of a Exec\_Component clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.17** void\* xml\_load\_exec\_instance (char \* *path*, const char \*\* *attr*)

Decode the arguments of a Exec\_Instance clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.18 void\* xml\_load\_hw\_component (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a HW\_Component clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.19 void\* xml\_load\_hw\_connection (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a HW\_Connection clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.20 void\* xml\_load\_hw\_group (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a Computing\_Group clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.21 void\* xml\_load\_hw\_instance (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a HW\_Instance clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.22 void\* xml\_load\_sw\_component (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a SW\_Component clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.23 void\* xml\_load\_sw\_instance (char \* *path*, const char \*\* *attr*)**

Decode the arguments of a SW instance clause.

**Parameters:**

*path* Clause path  
*name* Clause name

**Returns:**

the created struct

**7.20.1.24 void xml\_sw\_allocation\_string\_convert (struct sw\_allocation \* *allocation*)****7.20.1.25 void xml\_sw\_instance\_string\_convert (struct sw\_instance \* *instance*)****7.20.1.26 void xml\_sw\_task\_string\_convert (struct sw\_task \* *task*)****7.20.2 Variable Documentation****7.20.2.1 int xml\_info\_level****7.21 xml\_input.c File Reference**

```
#include "xml_input.h"  
#include "xml_list.h"  
#include "xml_if.h"  
#include <stdlib.h>  
#include <string.h>
```

## Functions

- char \* [xml\\_convert\\_path](#) (const char \*path)  
*Adjust a searching path by ensuring the path does not contain a ".".*
- struct [list\\_elem](#) \* [get\\_hw\\_component\\_header](#) ()  
*Return the list of HW components.*
- struct [list\\_elem](#) \* [get\\_hw\\_connection\\_header](#) ()  
*Return the list of HW connections.*
- struct [list\\_elem](#) \* [get\\_sw\\_component\\_header](#) ()  
*Return the list of SW components.*
- struct [list\\_elem](#) \* [get\\_sw\\_task\\_header](#) ()  
*Return the list of SW tasks.*
- struct [list\\_elem](#) \* [get\\_hw\\_instance\\_header](#) ()  
*Return the list of HW instances.*
- struct [list\\_elem](#) \* [get\\_sw\\_instance\\_header](#) ()  
*Return the list of SW instances.*
- struct [list\\_elem](#) \* [get\\_sw\\_allocation\\_header](#) ()  
*Return the list of SW allocations.*
- struct [list\\_elem](#) \* [get\\_hw\\_group\\_header](#) ()  
*Return the list of coomputing groups.*
- void [create\\_hw\\_component](#) (struct [hw\\_component](#) \*elem)  
*Insert a new HW component into the corresponding list.*
- struct [hw\\_component](#) \* [get\\_hw\\_component\\_by\\_name](#) (const char \*name)  
*Return the component indicated by the input name.*
- struct [hw\\_component](#) \* [get\\_hw\\_component\\_by\\_path](#) (const char \*path)  
*Return the component indicated by the input path.*
- void [create\\_hw\\_instance](#) (struct [hw\\_instance](#) \*elem)  
*Insert a new HW component into the corresponding list.*
- struct [hw\\_instance](#) \* [get\\_hw\\_instance\\_by\\_name](#) (const char \*name)  
*Return the component indicated by the input name.*
- struct [hw\\_instance](#) \* [get\\_hw\\_instance\\_by\\_path](#) (const char \*path)  
*Return the component indicated by the input path.*
- void [create\\_hw\\_connection](#) (struct [hw\\_connection](#) \*elem)  
*Insert a new HW component into the corresponding list.*

- void `create_hw_group` (struct `hw_group` \*elem)  
*Insert a new HW component into the corresponding list.*
- struct `hw_group` \* `get_hw_group_by_name` (const char \*name)  
*Return the component indicated by the input name.*
- struct `hw_group` \* `get_hw_group_by_path` (const char \*path)  
*Return the component indicated by the input path.*
- void `add_instance_to_hw_group` (struct `hw_group` \*group, struct `hw_instance` \*elem)  
*Add a HW instance to a computing group previously created.*
- void `create_sw_component` (struct `sw_component` \*elem)  
*Insert a new HW component into the corresponding list.*
- struct `sw_component` \* `get_sw_component_by_name` (const char \*name)  
*Return the component indicated by the input name.*
- struct `sw_component` \* `get_sw_component_by_path` (const char \*path)  
*Return the component indicated by the input path.*
- void `create_sw_instance` (struct `sw_instance` \*elem)  
*Insert a new HW component into the corresponding list.*
- struct `sw_instance` \* `get_sw_instance_by_name` (const char \*name)  
*Return the component indicated by the input name.*
- struct `sw_instance` \* `get_sw_instance_by_path` (const char \*path)  
*Return the component indicated by the input path.*
- void `create_sw_task` (struct `sw_task` \*elem)  
*Insert a new HW component into the corresponding list.*
- struct `sw_task` \* `get_sw_task_by_name` (const char \*name)  
*Return the component indicated by the input name.*
- struct `sw_task` \* `get_sw_task_by_path` (const char \*path)  
*Return the component indicated by the input path.*
- void `create_sw_allocation` (struct `sw_allocation` \*elem)  
*Insert a new HW component into the corresponding list.*

## Variables

- struct `list_elem` \* `hw_component_header` = 0
- struct `list_elem` \* `hw_instance_header` = 0
- struct `list_elem` \* `hw_connection_header` = 0
- struct `list_elem` \* `hw_group_header` = 0
- struct `list_elem` \* `sw_component_header` = 0



- struct [list\\_elem](#) \* [sw\\_instance\\_header](#) = 0
- struct [list\\_elem](#) \* [sw\\_task\\_header](#) = 0
- struct [list\\_elem](#) \* [sw\\_allocation\\_header](#) = 0

### 7.21.1 Function Documentation

#### 7.21.1.1 void add\_instance\_to\_hw\_group (struct hw\_group \* *group*, struct hw\_instance \* *elem*)

Add a HW instance to a computing group previously created.

**Parameters:**

- group* The computing group  
*elem* The component instance to be added

**Returns:**

#### 7.21.1.2 void create\_hw\_component (struct hw\_component \* *elem*)

Insert a new HW component into the corresponding list.

**Parameters:**

- elem* The new component

**Returns:**

#### 7.21.1.3 void create\_hw\_connection (struct hw\_connection \* *elem*)

Insert a new HW component into the corresponding list.

**Parameters:**

- elem* The new component

**Returns:**

#### 7.21.1.4 void create\_hw\_group (struct hw\_group \* *elem*)

Insert a new HW component into the corresponding list.

**Parameters:**

- elem* The new component

**Returns:**

**7.21.1.5 void create\_hw\_instance (struct hw\_instance \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.21.1.6 void create\_sw\_allocation (struct sw\_allocation \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.21.1.7 void create\_sw\_component (struct sw\_component \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.21.1.8 void create\_sw\_instance (struct sw\_instance \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.21.1.9 void create\_sw\_task (struct sw\_task \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:**

**7.21.1.10 struct hw\_component\* get\_hw\_component\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.21.1.11 struct hw\_component\* get\_hw\_component\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.21.1.12 struct list\_elem\* get\_hw\_component\_header ()** [read]

Return the list of HW components.

**Returns:**

The list pointer

**7.21.1.13 struct list\_elem\* get\_hw\_connection\_header ()** [read]

Return the list of HW connections.

**Returns:**

The list pointer

**7.21.1.14 struct hw\_group\* get\_hw\_group\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.21.1.15** `struct hw_group* get_hw_group_by_path (const char * path)` [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.21.1.16** `struct list_elem* get_hw_group_header ()` [read]

Return the list of coomputing groups.

**Returns:**

The list pointer

**7.21.1.17** `struct hw_instance* get_hw_instance_by_name (const char * name)` [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.21.1.18** `struct hw_instance* get_hw_instance_by_path (const char * path)` [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.21.1.19** `struct list_elem* get_hw_instance_header ()` [read]

Return the list of HW instances.

**Returns:**

The list pointer

**7.21.1.20 struct list\_elem\* get\_sw\_allocation\_header ()** [read]

Return the list of SW allocations.

**Returns:**

The list pointer

**7.21.1.21 struct sw\_component\* get\_sw\_component\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.21.1.22 struct sw\_component\* get\_sw\_component\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.21.1.23 struct list\_elem\* get\_sw\_component\_header ()** [read]

Return the list of SW components.

**Returns:**

The list pointer

**7.21.1.24 struct sw\_instance\* get\_sw\_instance\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.21.1.25 struct sw\_instance\* get\_sw\_instance\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.21.1.26 struct list\_elem\* get\_sw\_instance\_header ()** [read]

Return the list of SW instances.

**Returns:**

The list pointer

**7.21.1.27 struct sw\_task\* get\_sw\_task\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.21.1.28 struct sw\_task\* get\_sw\_task\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.21.1.29 struct list\_elem\* get\_sw\_task\_header ()** [read]

Return the list of SW tasks.

**Returns:**

The list pointer

### 7.21.1.30 char\* xml\_convert\_path (const char \* *path*)

Adjust a searching path by ensuring the path does not contain a ".0".

#### Parameters:

*path* The original path

#### Returns:

The fixed path

## 7.21.2 Variable Documentation

7.21.2.1 struct list\_elem\* hw\_component\_header = 0

7.21.2.2 struct list\_elem\* hw\_connection\_header = 0

7.21.2.3 struct list\_elem\* hw\_group\_header = 0

7.21.2.4 struct list\_elem\* hw\_instance\_header = 0

7.21.2.5 struct list\_elem\* sw\_allocation\_header = 0

7.21.2.6 struct list\_elem\* sw\_component\_header = 0

7.21.2.7 struct list\_elem\* sw\_instance\_header = 0

7.21.2.8 struct list\_elem\* sw\_task\_header = 0

## 7.22 xml\_input.h File Reference

### Functions

- struct list\_elem \* [get\\_hw\\_component\\_header](#) ()  
*Return the list of HW components.*
- struct list\_elem \* [get\\_hw\\_connection\\_header](#) ()  
*Return the list of HW connections.*
- struct list\_elem \* [get\\_sw\\_component\\_header](#) ()  
*Return the list of SW components.*
- struct list\_elem \* [get\\_sw\\_task\\_header](#) ()  
*Return the list of SW tasks.*
- struct list\_elem \* [get\\_hw\\_instance\\_header](#) ()

*Return the list of HW instances.*

- struct [list\\_elem](#) \* [get\\_sw\\_instance\\_header](#) ()  
*Return the list of SW instances.*
- struct [list\\_elem](#) \* [get\\_sw\\_allocation\\_header](#) ()  
*Return the list of SW allocations.*
- struct [list\\_elem](#) \* [get\\_hw\\_group\\_header](#) ()  
*Return the list of coomputing groups.*
- void [create\\_hw\\_component](#) (struct [hw\\_component](#) \*elem)  
*Insert a new HW component into the corresponding list.*
- struct [hw\\_component](#) \* [get\\_hw\\_component\\_by\\_name](#) (const char \*name)  
*Return the component indicated by the input name.*
- struct [hw\\_component](#) \* [get\\_hw\\_component\\_by\\_path](#) (const char \*path)  
*Return the component indicated by the input path.*
- void [create\\_hw\\_instance](#) (struct [hw\\_instance](#) \*elem)  
*Insert a new HW component into the corresponding list.*
- struct [hw\\_instance](#) \* [get\\_hw\\_instance\\_by\\_name](#) (const char \*name)  
*Return the component indicated by the input name.*
- struct [hw\\_instance](#) \* [get\\_hw\\_instance\\_by\\_path](#) (const char \*path)  
*Return the component indicated by the input path.*
- void [create\\_hw\\_connection](#) (struct [hw\\_connection](#) \*elem)  
*Insert a new HW component into the corresponding list.*
- void [create\\_hw\\_group](#) (struct [hw\\_group](#) \*elem)  
*Insert a new HW component into the corresponding list.*
- struct [hw\\_group](#) \* [get\\_hw\\_group\\_by\\_name](#) (const char \*name)  
*Return the component indicated by the input name.*
- struct [hw\\_group](#) \* [get\\_hw\\_group\\_by\\_path](#) (const char \*path)  
*Return the component indicated by the input path.*
- void [add\\_instance\\_to\\_hw\\_group](#) (struct [hw\\_group](#) \*group, struct [hw\\_instance](#) \*elem)  
*Add a HW instance to a computing group previously created.*
- void [create\\_sw\\_component](#) (struct [sw\\_component](#) \*elem)  
*Insert a new HW component into the corresponding list.*
- struct [sw\\_component](#) \* [get\\_sw\\_component\\_by\\_name](#) (const char \*name)  
*Return the component indicated by the input name.*



- struct `sw_component` \* `get_sw_component_by_path` (const char \*path)  
*Return the component indicated by the input path.*
- void `create_sw_instance` (struct `sw_instance` \*elem)  
*Insert a new HW component into the corresponding list.*
- struct `sw_instance` \* `get_sw_instance_by_name` (const char \*name)  
*Return the component indicated by the input name.*
- struct `sw_instance` \* `get_sw_instance_by_path` (const char \*path)  
*Return the component indicated by the input path.*
- void `create_sw_task` (struct `sw_task` \*elem)  
*Insert a new HW component into the corresponding list.*
- struct `sw_task` \* `get_sw_task_by_name` (const char \*name)  
*Return the component indicated by the input name.*
- struct `sw_task` \* `get_sw_task_by_path` (const char \*path)  
*Return the component indicated by the input path.*
- void `create_sw_allocation` (struct `sw_allocation` \*elem)  
*Insert a new HW component into the corresponding list.*

### 7.22.1 Function Documentation

#### 7.22.1.1 void add\_instance\_to\_hw\_group (struct hw\_group \*group, struct hw\_instance \* elem)

Add a HW instance to a computing group previously created.

##### Parameters:

- group* The computing group  
*elem* The component instance to be added

##### Returns:

#### 7.22.1.2 void create\_hw\_component (struct hw\_component \* elem)

Insert a new HW component into the corresponding list.

##### Parameters:

- elem* The new component

##### Returns:

**7.22.1.3 void create\_hw\_connection (struct hw\_connection \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.22.1.4 void create\_hw\_group (struct hw\_group \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.22.1.5 void create\_hw\_instance (struct hw\_instance \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.22.1.6 void create\_sw\_allocation (struct sw\_allocation \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.22.1.7 void create\_sw\_component (struct sw\_component \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:**

**7.22.1.8 void create\_sw\_instance (struct sw\_instance \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.22.1.9 void create\_sw\_task (struct sw\_task \* *elem*)**

Insert a new HW component into the corresponding list.

**Parameters:**

*elem* The new component

**Returns:****7.22.1.10 struct hw\_component\* get\_hw\_component\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.22.1.11 struct hw\_component\* get\_hw\_component\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.22.1.12 struct list\_elem\* get\_hw\_component\_header ()** [read]

Return the list of HW components.

**Returns:**

The list pointer

**7.22.1.13 struct list\_elem\* get\_hw\_connection\_header () [read]**

Return the list of HW connections.

**Returns:**

The list pointer

**7.22.1.14 struct hw\_group\* get\_hw\_group\_by\_name (const char \* *name*) [read]**

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.22.1.15 struct hw\_group\* get\_hw\_group\_by\_path (const char \* *path*) [read]**

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.22.1.16 struct list\_elem\* get\_hw\_group\_header () [read]**

Return the list of coomputing groups.

**Returns:**

The list pointer

**7.22.1.17 struct hw\_instance\* get\_hw\_instance\_by\_name (const char \* *name*) [read]**

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.22.1.18 struct hw\_instance\* get\_hw\_instance\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.22.1.19 struct list\_elem\* get\_hw\_instance\_header ()** [read]

Return the list of HW instances.

**Returns:**

The list pointer

**7.22.1.20 struct list\_elem\* get\_sw\_allocation\_header ()** [read]

Return the list of SW allocations.

**Returns:**

The list pointer

**7.22.1.21 struct sw\_component\* get\_sw\_component\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.22.1.22 struct sw\_component\* get\_sw\_component\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.22.1.23 struct list\_elem\* get\_sw\_component\_header ()** [read]

Return the list of SW components.

**Returns:**

The list pointer

**7.22.1.24 struct sw\_instance\* get\_sw\_instance\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.22.1.25 struct sw\_instance\* get\_sw\_instance\_by\_path (const char \* *path*)** [read]

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.22.1.26 struct list\_elem\* get\_sw\_instance\_header ()** [read]

Return the list of SW instances.

**Returns:**

The list pointer

**7.22.1.27 struct sw\_task\* get\_sw\_task\_by\_name (const char \* *name*)** [read]

Return the component indicated by the input name.

**Parameters:**

*name* The component name

**Returns:**

the component

**7.22.1.28 struct sw\_task\* get\_sw\_task\_by\_path (const char \*path) [read]**

Return the component indicated by the input path.

**Parameters:**

*name* The component path

**Returns:**

the component

**7.22.1.29 struct list\_elem\* get\_sw\_task\_header () [read]**

Return the list of SW tasks.

**Returns:**

The list pointer

**7.23 xml\_list.c File Reference**

```
#include "xml_list.h"
```

```
#include <stdlib.h>
```

**Functions**

- int [xml\\_list\\_size](#) (struct [list\\_elem](#) \*\*header)  
*Return the size of the list.*
- void [xml\\_list\\_add\\_element](#) (struct [list\\_elem](#) \*\*header, void \*data)  
*Add an new element to the end of the list.*
- void \* [xml\\_list\\_last\\_element](#) (struct [list\\_elem](#) \*header)  
*Peek the last element of the list.*

**7.23.1 Function Documentation****7.23.1.1 void xml\_list\_add\_element (struct list\_elem \*\* header, void \* data)**

Add an new element to the end of the list.

**Parameters:**

*header* Pointer to the list (pointer to the first element)

*data* Pointer to the new data to be stored

**Returns:**

### 7.23.1.2 void\* xml\_list\_last\_element (struct list\_elem \* header)

Peek the last element of the list.

#### Parameters:

**header** Pointer to the list (pointer to the first element)  
**\return** pointer to the last element of the list

### 7.23.1.3 int xml\_list\_size (struct list\_elem \*\* header)

Return the size of the list.

#### Parameters:

**header** Pointer to the list (pointer to the first element)

#### Returns:

The list size

## 7.24 xml\_list.h File Reference

### Data Structures

- struct [list\\_elem](#)

### Functions

- int [xml\\_list\\_size](#) (struct [list\\_elem](#) \*\*)  
*Return the size of the list.*
- void [xml\\_list\\_add\\_element](#) (struct [list\\_elem](#) \*\*, void \*)  
*Add an new element to the end of the list.*
- void \* [xml\\_list\\_last\\_element](#) (struct [list\\_elem](#) \*)  
*Peek the last element of the list.*

### 7.24.1 Function Documentation

#### 7.24.1.1 void xml\_list\_add\_element (struct list\_elem \*\* header, void \* data)

Add an new element to the end of the list.

#### Parameters:

**header** Pointer to the list (pointer to the first element)  
**data** Pointer to the new data to be stored

#### Returns:



#### 7.24.1.2 void\* xml\_list\_last\_element (struct list\_elem \* header)

Peek the last element of the list.

##### Parameters:

*header* Pointer to the list (pointer to the first element)  
\return pointer to the last element of the list

#### 7.24.1.3 int xml\_list\_size (struct list\_elem \*\* header)

Return the size of the list.

##### Parameters:

*header* Pointer to the list (pointer to the first element)

##### Returns:

The list size

## 7.25 xml\_main.c File Reference

```
#include "xml_main.h"
#include "xml_configuration_file.h"
#include "xml_hierarchy.h"
#include "xml_if.h"
#include <stdio.h>
#include <string.h>
#include "expat.h"
```

### Functions

- int [read\\_xml\\_file](#) (char \*file)  
*Function read the XML System Description file.*

### Variables

- int [xml\\_info\\_level](#) = 0
- int [xml\\_has\\_platform\\_description](#) = 0

### 7.25.1 Function Documentation

#### 7.25.1.1 int read\_xml\_file (char \* file)

Function read the XML System Description file.

##### Parameters:

*file* Name of the file

**Returns:**

0 on success, otherwise error

**7.25.2 Variable Documentation****7.25.2.1 int xml\_has\_platform\_description = 0****7.25.2.2 int xml\_info\_level = 0****7.26 xml\_main.h File Reference****Functions**

- int [read\\_xml\\_file](#) (char \*file)  
*Function read the XML System Description file.*

**7.26.1 Function Documentation****7.26.1.1 int read\_xml\_file (char \*file)**

Function read the XML System Description file.

**Parameters:**

*file* Name of the file

**Returns:**

0 on success, otherwise error

**7.27 xml\_obtain\_metrics.cpp File Reference**

```
#include "xml_if.h"
#include "xml_input.h"
#include "xml_list.h"
#include <algorithm>
#include "xml_obtain_metrics.h"
#include "scuc_noc_sim_if.h"
```

**Defines**

- #define [GET\\_INSTANCE\\_PARAMETER](#)(instance, param)

## Functions

- long double [xml\\_get\\_tlm\\_power\\_consumption](#) (UC\_tlm\_statistics \*statistics, struct [hw\\_instance](#) \*instance)
- long double [calculate\\_latency](#) ()
- unsigned long int [calculate\\_ec](#) ()
- unsigned long int [calculate\\_ic](#) ()
- unsigned long int [calculate\\_mips](#) ()
- long double [calculate\\_hr](#) ()
- unsigned long int [calculate\\_msc](#) ()
- long double [calculate\\_amat](#) ()
- int [obtain\\_xml\\_metrics](#) (MetricReader &mr)

### 7.27.1 Define Documentation

#### 7.27.1.1 #define GET\_INSTANCE\_PARAMETER(instance, param)

Value:

```
(instance->param != NULL) ? atoi_null(instance->param) : \
((instance->component->param != NULL) ? atoi_null(instance->component->param) : 0)
```

### 7.27.2 Function Documentation

#### 7.27.2.1 long double calculate\_amat ()

#### 7.27.2.2 unsigned long int calculate\_ec ()

#### 7.27.2.3 long double calculate\_hr ()

#### 7.27.2.4 unsigned long int calculate\_ic ()

#### 7.27.2.5 long double calculate\_latency ()

#### 7.27.2.6 unsigned long int calculate\_mips ()

#### 7.27.2.7 unsigned long int calculate\_msc ()

#### 7.27.2.8 int obtain\_xml\_metrics (MetricReader & mr)

#### 7.27.2.9 long double xml\_get\_tlm\_power\_consumption (UC\_tlm\_statistics \* *statistics*, struct hw\_instance \* *instance*)

## 7.28 xml\_obtain\_metrics.h File Reference

```
#include "sc_scope.h"  
#include "xml_main.h"  
#include "uc_load_xml.h"
```

### Functions

- int [obtain\\_xml\\_metrics](#) ([MetricReader](#) &mr)

#### 7.28.1 Function Documentation

##### 7.28.1.1 int obtain\_xml\_metrics ([MetricReader](#) & *mr*)

## Index

- ~ExpatXMLParser
  - expatmm::ExpatXMLParser, [5](#)
- ~Metric
  - Metric, [16](#)
- ~MetricReader
  - MetricReader, [21](#)
- \_xml\_intermediate\_function
  - uc\_create\_xml\_platform.cpp, [34](#)
- \_xml\_taskload\_function
  - uc\_create\_xml\_platform.cpp, [34](#)
- activate\_backtrace
  - uc\_load\_xml.cpp, [38](#)
- activation\_type
  - hw\_component, [9](#)
  - hw\_connection, [10](#)
  - hw\_instance, [12](#)
- add\_instance\_to\_hw\_group
  - xml\_input.c, [64](#)
  - xml\_input.h, [72](#)
- allocate\_sw\_tasks
  - uc\_create\_xml\_platform.cpp, [34](#)
  - uc\_create\_xml\_platform.h, [36](#)
- area
  - hw\_component, [9](#)
  - hw\_instance, [13](#)
- args
  - sw\_allocation, [24](#)
  - xml\_basic\_info, [27](#)
- atoi\_null
  - xml\_if.c, [52](#)
  - xml\_if.h, [57](#)
- calculate\_amat
  - xml\_obtain\_metrics.cpp, [82](#)
- calculate\_ec
  - xml\_obtain\_metrics.cpp, [82](#)
- calculate\_hr
  - xml\_obtain\_metrics.cpp, [82](#)
- calculate\_ic
  - xml\_obtain\_metrics.cpp, [82](#)
- calculate\_latency
  - xml\_obtain\_metrics.cpp, [82](#)
- calculate\_mips
  - xml\_obtain\_metrics.cpp, [82](#)
- calculate\_msc
  - xml\_obtain\_metrics.cpp, [82](#)
- CDataEnd
  - expatmm::ExpatXMLParser, [6, 8](#)
- CDataStart
  - expatmm::ExpatXMLParser, [6, 8](#)
- CharacterData
  - expatmm::ExpatXMLParser, [6, 7](#)
- children\_header
  - xml\_basic\_info, [27](#)
- class\_name
  - hw\_component, [9](#)
  - hw\_instance, [12](#)
  - sw\_task, [26](#)
- CommentData
  - expatmm::ExpatXMLParser, [6, 7](#)
- component
  - hw\_instance, [13](#)
  - sw\_instance, [25](#)
- component\_name
  - hw\_instance, [13](#)
  - sw\_instance, [25](#)
- compute\_time
  - sw\_task, [26](#)
- connect\_hw\_components
  - uc\_create\_xml\_platform.cpp, [34](#)
  - uc\_create\_xml\_platform.h, [36](#)
- connections
  - hw\_instance, [13](#)
- contain
  - hw\_instance, [13](#)
- create\_hw\_component
  - xml\_input.c, [64](#)
  - xml\_input.h, [72](#)
- create\_hw\_connection
  - xml\_input.c, [64](#)
  - xml\_input.h, [72](#)
- create\_hw\_group
  - xml\_input.c, [64](#)
  - xml\_input.h, [73](#)
- create\_hw\_instance
  - xml\_input.c, [64](#)
  - xml\_input.h, [73](#)
- create\_sw\_allocation
  - xml\_input.c, [65](#)
  - xml\_input.h, [73](#)
- create\_sw\_component
  - xml\_input.c, [65](#)
  - xml\_input.h, [73](#)
- create\_sw\_instance
  - xml\_input.c, [65](#)
  - xml\_input.h, [73](#)
- create\_sw\_task
  - xml\_input.c, [65](#)
  - xml\_input.h, [74](#)
- data\_size

- sw\_task, 27
- debug\_level
  - xml\_scope\_simulation\_info\_t, 28
- DefaultHandler
  - expatmm::ExpatXMLParser, 6, 8
- destroy\_objects
  - uc\_load\_xml.cpp, 37
  - uc\_load\_xml.h, 39
- elem
  - list\_elem, 14
- EndElement
  - expatmm::ExpatXMLParser, 6, 7
  - MetricReader, 23
- executable\_name
  - uc\_create\_xml\_platform.cpp, 36
  - uc\_load\_xml.cpp, 38
- expatmm, 4
  - getExpatMMVersion, 4
- expatmm-libdef.h, 29
- ExpatMM-version.cpp, 29
- expatmm.h, 29
  - EXPATMM\_LIBRARY\_MAJOR, 29
  - EXPATMM\_LIBRARY\_MINOR, 29
  - EXPATMM\_LIBRARY\_REVISION, 29
  - EXPATMM\_LIBRARY\_VERSION, 29
- expatmm/ExpatXMLParser.h
  - XML\_CHUNK\_SIZE, 31
- expatmm::ExpatXMLParser, 4
  - ~ExpatXMLParser, 5
  - CDataEnd, 6, 8
  - CDataStart, 6, 8
  - CharacterData, 6, 7
  - CommentData, 6, 7
  - DefaultHandler, 6, 8
  - EndElement, 6, 7
  - ExpatXMLParser, 5
  - getBlockSize, 6, 7
  - getBuffer, 6, 7
  - getLastError, 7, 8
  - getStatus, 7, 8
  - Parse, 6, 8
  - ProcessingInstruction, 6, 7
  - read\_block, 6, 7
  - Ready, 7, 8
  - setLastError, 6, 7
  - setReadiness, 6, 7
  - setStatus, 6, 7
  - StartElement, 6, 7
- EXPATMM\_LIBRARY\_MAJOR
  - expatmm.h, 29
- EXPATMM\_LIBRARY\_MINOR
  - expatmm.h, 29
- EXPATMM\_LIBRARY\_REVISION
  - expatmm.h, 29
- expatmm.h, 29
  - expatmm.h, 29
- EXPATMM\_LIBRARY\_VERSION
  - expatmm.h, 29
- ExpatXMLParser
  - expatmm::ExpatXMLParser, 5
- ExpatXMLParser.cpp, 30
- ExpatXMLParser.h, 30
- file
  - sw\_task, 26
- fin\_time
  - sw\_task, 27
- freq
  - hw\_component, 9
  - hw\_instance, 12
- function
  - sw\_task, 26
- get\_dynamic\_function
  - uc\_create\_xml\_platform.cpp, 34
- get\_hw\_component\_by\_name
  - xml\_input.c, 65
  - xml\_input.h, 74
- get\_hw\_component\_by\_path
  - xml\_input.c, 66
  - xml\_input.h, 74
- get\_hw\_component\_header
  - xml\_input.c, 66
  - xml\_input.h, 74
- get\_hw\_connection\_header
  - xml\_input.c, 66
  - xml\_input.h, 74
- get\_hw\_group\_by\_name
  - xml\_input.c, 66
  - xml\_input.h, 75
- get\_hw\_group\_by\_path
  - xml\_input.c, 66
  - xml\_input.h, 75
- get\_hw\_group\_header
  - xml\_input.c, 67
  - xml\_input.h, 75
- get\_hw\_instance\_by\_name
  - xml\_input.c, 67
  - xml\_input.h, 75
- get\_hw\_instance\_by\_path
  - xml\_input.c, 67
  - xml\_input.h, 75
- get\_hw\_instance\_header
  - xml\_input.c, 67
  - xml\_input.h, 76
- GET\_INSTANCE\_PARAMETER
  - xml\_obtain\_metrics.cpp, 82
- get\_static\_function
  - uc\_create\_xml\_platform.cpp, 34

- get\_sw\_allocation\_header
  - xml\_input.c, 67
  - xml\_input.h, 76
- get\_sw\_component\_by\_name
  - xml\_input.c, 68
  - xml\_input.h, 76
- get\_sw\_component\_by\_path
  - xml\_input.c, 68
  - xml\_input.h, 76
- get\_sw\_component\_header
  - xml\_input.c, 68
  - xml\_input.h, 76
- get\_sw\_instance\_by\_name
  - xml\_input.c, 68
  - xml\_input.h, 77
- get\_sw\_instance\_by\_path
  - xml\_input.c, 68
  - xml\_input.h, 77
- get\_sw\_instance\_header
  - xml\_input.c, 69
  - xml\_input.h, 77
- get\_sw\_task\_by\_name
  - xml\_input.c, 69
  - xml\_input.h, 77
- get\_sw\_task\_by\_path
  - xml\_input.c, 69
  - xml\_input.h, 77
- get\_sw\_task\_header
  - xml\_input.c, 69
  - xml\_input.h, 78
- getBlockSize
  - expatmm::ExpatXMLParser, 6, 7
- getBuffer
  - expatmm::ExpatXMLParser, 6, 7
- getExpatMMVersion
  - expatmm, 4
- getFloatValue
  - Metric, 17
- getIntegerValue
  - Metric, 17
- getLastError
  - expatmm::ExpatXMLParser, 7, 8
- getMetric
  - MetricReader, 22
- getName
  - Metric, 17
- getStatus
  - expatmm::ExpatXMLParser, 7, 8
- getType
  - Metric, 17
- getUnit
  - Metric, 17
- gui\_connector
  - uc\_load\_xml.h, 38
- uc\_load\_xml.h, 40
- hw\_component, 8
  - activation\_type, 9
  - area, 9
  - class\_name, 9
  - freq, 9
  - latency, 9
  - mean\_power, 9
  - mem\_size, 9
  - name, 9
  - path, 9
  - read\_energy, 9
  - read\_size\_energy, 9
  - type, 9
  - type\_specific\_1, 9
  - type\_specific\_2, 9
  - width, 9
  - write\_energy, 9
  - write\_size\_energy, 9
- hw\_component\_header
  - xml\_input.c, 70
- hw\_connection, 10
  - activation\_type, 10
  - instance, 11
  - instance\_name, 11
  - irq, 11
  - latency, 10
  - local\_id, 11
  - mem\_size, 10
  - name, 10
  - offset, 11
  - path, 10
  - port, 11
  - rec\_irq, 11
  - speed, 10
  - start\_addr, 10
  - type\_specific\_1, 10
  - type\_specific\_2, 10
- hw\_connection\_header
  - xml\_input.c, 70
- hw\_group, 11
  - hw\_names, 11
  - list, 11
  - name, 11
  - path, 11
  - sw\_allocation, 24
  - sw\_instance, 26
- hw\_group\_header
  - xml\_input.c, 70
- hw\_instance, 12
  - activation\_type, 12
  - area, 13
  - class\_name, 12

- component, 13
- component\_name, 13
- connections, 13
- contain, 13
- freq, 12
- irq, 13
- latency, 12
- local\_id, 13
- mean\_power, 13
- mem\_size, 13
- name, 12
- offset, 13
- path, 12
- read\_energy, 13
- read\_size\_energy, 13
- scope\_data, 13
- start\_addr, 13
- type, 12
- type\_specific\_1, 13
- type\_specific\_2, 13
- width, 13
- write\_energy, 13
- write\_size\_energy, 13
- hw\_instance\_header
  - xml\_input.c, 70
- hw\_names
  - hw\_group, 11
- hw\_resource
  - sw\_allocation, 24
  - sw\_instance, 26
- index
  - xml\_basic\_info, 27
- init\_index
  - xml\_basic\_info, 27
- init\_time
  - sw\_task, 27
- instance
  - hw\_connection, 11
- instance\_name
  - hw\_connection, 11
- instantiate\_hw\_components
  - uc\_create\_xml\_platform.cpp, 35
  - uc\_create\_xml\_platform.h, 36
- instantiate\_sw\_components
  - uc\_create\_xml\_platform.cpp, 35
  - uc\_create\_xml\_platform.h, 36
- irq
  - hw\_connection, 11
  - hw\_instance, 13
- isMetricLeft
  - MetricReader, 22
- latency
  - hw\_component, 9
  - hw\_connection, 10
  - hw\_instance, 12
- list
  - hw\_group, 11
- list\_elem, 14
  - elem, 14
  - next, 14
- load\_task\_info
  - uc\_create\_xml\_platform.cpp, 35
- load\_taskload\_info
  - uc\_create\_xml\_platform.cpp, 35
- load\_xml\_configuration\_file
  - uc\_load\_xml.cpp, 37
  - uc\_load\_xml.h, 39
- load\_xml\_platform\_file
  - uc\_load\_xml.cpp, 38
  - uc\_load\_xml.h, 40
- local\_id
  - hw\_connection, 11
  - hw\_instance, 13
- mean\_power
  - hw\_component, 9
  - hw\_instance, 13
- mem\_size
  - hw\_component, 9
  - hw\_connection, 10
  - hw\_instance, 13
- Metric, 14
  - ~Metric, 16
  - getFloatValue, 17
  - getIntegerValue, 17
  - getName, 17
  - getType, 17
  - getUnit, 17
  - Metric, 16
  - multiplier, 17
  - nameToLower, 17
  - operator<<, 19
  - operator=, 18
  - setName, 17
  - setType, 18
  - setUnit, 18
  - setValue, 18
- metric.cpp, 31
  - operator<<, 31
- metric.h, 31
- MetricReader, 19
  - ~MetricReader, 21
  - EndElement, 23
  - getMetric, 22
  - isMetricLeft, 22
  - MetricReader, 21



- read\_block, [22](#)
- setMetric, [22](#)
- size, [22](#)
- StartElement, [22](#)
- writeXMLErrorFile, [21](#)
- writeXMLFile, [21](#)
- multiplier
  - Metric, [17](#)
- name
  - hw\_component, [9](#)
  - hw\_connection, [10](#)
  - hw\_group, [11](#)
  - hw\_instance, [12](#)
  - sw\_allocation, [24](#)
  - sw\_component, [25](#)
  - sw\_instance, [25](#)
  - sw\_task, [26](#)
  - xml\_basic\_info, [27](#)
  - xml\_parameter\_info, [28](#)
- nameToLower
  - Metric, [17](#)
- next
  - list\_elem, [14](#)
- obtain\_xml\_metrics
  - uc\_load\_xml.h, [40](#)
  - xml\_obtain\_metrics.cpp, [82](#)
  - xml\_obtain\_metrics.h, [83](#)
- offset
  - hw\_connection, [11](#)
  - hw\_instance, [13](#)
  - sw\_allocation, [24](#)
  - sw\_instance, [26](#)
- operator<<
  - Metric, [19](#)
  - metric.cpp, [31](#)
- operator=
  - Metric, [18](#)
- os\_name
  - sw\_allocation, [24](#)
- parent
  - xml\_basic\_info, [27](#)
- Parse
  - expatmm::ExpatXMLParser, [6, 8](#)
- parse.cpp, [32](#)
- parse.h, [32](#)
- path
  - hw\_component, [9](#)
  - hw\_connection, [10](#)
  - hw\_group, [11](#)
  - hw\_instance, [12](#)
  - sw\_allocation, [24](#)
  - sw\_component, [25](#)
  - sw\_instance, [25](#)
  - sw\_task, [26](#)
- period
  - sw\_task, [27](#)
- policy
  - sw\_allocation, [24](#)
- port
  - hw\_connection, [11](#)
- prev\_xml\_parameter\_info
  - xml\_configuration\_file.c, [42](#)
- priority
  - sw\_allocation, [24](#)
- ProcessingInstruction
  - expatmm::ExpatXMLParser, [6, 7](#)
- read\_block
  - expatmm::ExpatXMLParser, [6, 7](#)
  - MetricReader, [22](#)
- read\_energy
  - hw\_component, [9](#)
  - hw\_instance, [13](#)
- read\_size\_energy
  - hw\_component, [9](#)
  - hw\_instance, [13](#)
- read\_xml\_file
  - xml\_main.c, [80](#)
  - xml\_main.h, [81](#)
- Ready
  - expatmm::ExpatXMLParser, [7, 8](#)
- rec\_irq
  - hw\_connection, [11](#)
- repeat
  - xml\_basic\_info, [27](#)
- resource\_name
  - sw\_allocation, [24](#)
  - sw\_instance, [25](#)
- rtos\_list
  - uc\_load\_xml.cpp, [38](#)
  - uc\_load\_xml.h, [40](#)
- sc\_main
  - uc\_load\_xml.cpp, [38](#)
  - uc\_load\_xml.h, [40](#)
- scope\_data
  - hw\_instance, [13](#)
  - sw\_allocation, [24](#)
  - sw\_instance, [26](#)
- setLastError
  - expatmm::ExpatXMLParser, [6, 7](#)
- setMetric
  - MetricReader, [22](#)
- setName
  - Metric, [17](#)

- setReadiness
  - expatmm::ExpatXMLParser, 6, 7
- setStatus
  - expatmm::ExpatXMLParser, 6, 7
- setType
  - Metric, 18
- setUnit
  - Metric, 18
- setValue
  - Metric, 18
- sim\_time\_unit
  - xml\_scope\_simulation\_info\_t, 28
- simulation\_time
  - xml\_scope\_simulation\_info\_t, 28
- simulator\_list
  - uc\_load\_xml.cpp, 38
  - uc\_load\_xml.h, 40
- size
  - MetricReader, 22
- speed
  - hw\_connection, 10
- src/ExpatXMLParser.h
  - XML\_CHUNK\_SIZE, 30
- start\_addr
  - hw\_connection, 10
  - hw\_instance, 13
- StartElement
  - expatmm::ExpatXMLParser, 6, 7
  - MetricReader, 22
- strcasecmp\_null
  - xml\_if.c, 53
  - xml\_if.h, 57
- sw\_allocation, 23
  - args, 24
  - hw\_group, 24
  - hw\_resource, 24
  - name, 24
  - offset, 24
  - os\_name, 24
  - path, 24
  - policy, 24
  - priority, 24
  - resource\_name, 24
  - scope\_data, 24
  - sw\_resource, 24
  - task, 24
  - task\_name, 24
- sw\_allocation\_header
  - xml\_input.c, 70
- sw\_component, 24
  - name, 25
  - path, 25
  - type, 25
- sw\_component\_header
  - xml\_input.c, 70
- sw\_instance, 25
  - component, 25
  - component\_name, 25
  - hw\_group, 26
  - hw\_resource, 26
  - name, 25
  - offset, 26
  - path, 25
  - resource\_name, 25
  - scope\_data, 26
  - type, 25
- sw\_instance\_header
  - xml\_input.c, 70
- sw\_resource
  - sw\_allocation, 24
- sw\_task, 26
  - class\_name, 26
  - compute\_time, 26
  - data\_size, 27
  - file, 26
  - fin\_time, 27
  - function, 26
  - init\_time, 27
  - name, 26
  - path, 26
  - period, 27
  - type, 26
- sw\_task\_header
  - xml\_input.c, 70
- system\_buses
  - uc\_create\_xml\_platform.cpp, 36
- task
  - sw\_allocation, 24
- task\_name
  - sw\_allocation, 24
- type
  - hw\_component, 9
  - hw\_instance, 12
  - sw\_component, 25
  - sw\_instance, 25
  - sw\_task, 26
- type\_specific\_1
  - hw\_component, 9
  - hw\_connection, 10
  - hw\_instance, 13
- type\_specific\_2
  - hw\_component, 9
  - hw\_connection, 10
  - hw\_instance, 13
- uc\_create\_xml\_platform.cpp, 32
  - \_xml\_intermediate\_function, 34

- [\\_xml\\_taskload\\_function](#), 34
  - [allocate\\_sw\\_tasks](#), 34
  - [connect\\_hw\\_components](#), 34
  - [executable\\_name](#), 36
  - [get\\_dynamic\\_function](#), 34
  - [get\\_static\\_function](#), 34
  - [instanciate\\_hw\\_components](#), 35
  - [instanciate\\_sw\\_components](#), 35
  - [load\\_task\\_info](#), 35
  - [load\\_taskload\\_info](#), 35
  - [system\\_buses](#), 36
  - [uc\\_segment\\_time](#), 36
- [uc\\_create\\_xml\\_platform.h](#), 36
  - [allocate\\_sw\\_tasks](#), 36
  - [connect\\_hw\\_components](#), 36
  - [instanciate\\_hw\\_components](#), 36
  - [instanciate\\_sw\\_components](#), 36
- [uc\\_load\\_xml.cpp](#), 37
  - [activate\\_backtrace](#), 38
  - [destroy\\_objects](#), 37
  - [executable\\_name](#), 38
  - [gui\\_connector](#), 38
  - [load\\_xml\\_configuration\\_file](#), 37
  - [load\\_xml\\_platform\\_file](#), 38
  - [rtos\\_list](#), 38
  - [sc\\_main](#), 38
  - [simulator\\_list](#), 38
  - [usage](#), 38
  - [xml\\_scope\\_simulation\\_info](#), 38
- [uc\\_load\\_xml.h](#), 39
  - [destroy\\_objects](#), 39
  - [gui\\_connector](#), 40
  - [load\\_xml\\_configuration\\_file](#), 39
  - [load\\_xml\\_platform\\_file](#), 40
  - [obtain\\_xml\\_metrics](#), 40
  - [rtos\\_list](#), 40
  - [sc\\_main](#), 40
  - [simulator\\_list](#), 40
  - [usage](#), 40
  - [xml\\_scope\\_simulation\\_info](#), 40
- [uc\\_segment\\_time](#)
  - [uc\\_create\\_xml\\_platform.cpp](#), 36
- [usage](#)
  - [uc\\_load\\_xml.cpp](#), 38
  - [uc\\_load\\_xml.h](#), 40
- [value](#)
  - [xml\\_parameter\\_info](#), 28
- [warnings](#)
  - [xml\\_scope\\_simulation\\_info\\_t](#), 28
- [width](#)
  - [hw\\_component](#), 9
  - [hw\\_instance](#), 13
- [write\\_energy](#)
  - [hw\\_component](#), 9
  - [hw\\_instance](#), 13
- [write\\_size\\_energy](#)
  - [hw\\_component](#), 9
  - [hw\\_instance](#), 13
- [writeXMLErrorFile](#)
  - [MetricReader](#), 21
- [writeXMLFile](#)
  - [MetricReader](#), 21
- [xml\\_add\\_hierarchy](#)
  - [xml\\_hierarchy.c](#), 44
  - [xml\\_hierarchy.h](#), 49
- [xml\\_add\\_parameter](#)
  - [xml\\_configuration\\_file.c](#), 41
  - [xml\\_configuration\\_file.h](#), 42
- [xml\\_add\\_process\\_argument](#)
  - [xml\\_if.c](#), 53
  - [xml\\_if.h](#), 58
- [xml\\_add\\_to\\_hw\\_group](#)
  - [xml\\_if.c](#), 53
  - [xml\\_if.h](#), 58
- [xml\\_basic\\_info](#), 27
  - [args](#), 27
  - [children\\_header](#), 27
  - [index](#), 27
  - [init\\_index](#), 27
  - [name](#), 27
  - [parent](#), 27
  - [repeat](#), 27
- [xml\\_check\\_instance](#)
  - [xml\\_hierarchy.c](#), 45
  - [xml\\_hierarchy.h](#), 49
- [xml\\_check\\_name](#)
  - [xml\\_hierarchy.c](#), 45
  - [xml\\_hierarchy.h](#), 49
- [XML\\_CHUNK\\_SIZE](#)
  - [expatmm/ExpatXMLParser.h](#), 31
  - [src/ExpatXMLParser.h](#), 30
- [xml\\_close\\_declare](#)
  - [xml\\_if.c](#), 53
  - [xml\\_if.h](#), 58
- [xml\\_config\\_parameter\\_header](#)
  - [xml\\_configuration\\_file.c](#), 42
- [xml\\_configuration\\_file.c](#), 41
  - [prev\\_xml\\_parameter\\_info](#), 42
  - [xml\\_add\\_parameter](#), 41
  - [xml\\_config\\_parameter\\_header](#), 42
  - [xml\\_end\\_parameter](#), 41
  - [xml\\_load\\_configuration\\_parameter](#), 42
- [xml\\_configuration\\_file.h](#), 42
  - [xml\\_add\\_parameter](#), 42
  - [xml\\_end\\_parameter](#), 43

- xml\_load\_configuration\_parameter, 43
- xml\_connect\_hw\_connection
  - xml\_if.h, 58
- xml\_connect\_hw\_group
  - xml\_if.h, 58
- xml\_connect\_hw\_instance
  - xml\_if.h, 58
- xml\_connect\_sw\_allocation
  - xml\_if.h, 59
- xml\_connect\_sw\_instance
  - xml\_if.h, 59
- xml\_convert\_path
  - xml\_input.c, 69
- xml\_declare\_function
  - xml\_if.c, 53
  - xml\_if.h, 59
- xml\_decrease\_path
  - xml\_hierarchy.c, 45
  - xml\_hierarchy.h, 49
- xml\_display\_hierarchy
  - xml\_hierarchy.c, 45
  - xml\_hierarchy.h, 49
- xml\_display\_hierarchy\_node
  - xml\_hierarchy.c, 46
- xml\_end\_parameter
  - xml\_configuration\_file.c, 41
  - xml\_configuration\_file.h, 43
- xml\_get\_tlm\_power\_consumption
  - xml\_obtain\_metrics.cpp, 82
- xml\_has\_platform\_description
  - xml\_main.c, 81
- xml\_hierarchy.c, 43
  - xml\_add\_hierarchy, 44
  - xml\_check\_instance, 45
  - xml\_check\_name, 45
  - xml\_decrease\_path, 45
  - xml\_display\_hierarchy, 45
  - xml\_display\_hierarchy\_node, 46
  - xml\_hierarchy\_current, 47
  - xml\_hierarchy\_header, 47
  - xml\_increase\_path, 46
  - xml\_indexes, 48
  - xml\_search\_path, 46
  - xml\_select\_implementation, 46
  - xml\_select\_implementation\_array, 48
  - xml\_simulation\_parameters, 47
  - xml\_string\_index\_convert, 47
  - xml\_up\_hierarchy, 47
- xml\_hierarchy.h, 48
  - xml\_add\_hierarchy, 49
  - xml\_check\_instance, 49
  - xml\_check\_name, 49
  - xml\_decrease\_path, 49
  - xml\_display\_hierarchy, 49
  - xml\_increase\_path, 50
  - xml\_search\_path, 50
  - xml\_select\_implementation, 50
  - xml\_simulation\_parameters, 50
  - xml\_string\_index\_convert, 51
  - xml\_up\_hierarchy, 51
- xml\_hierarchy\_current
  - xml\_hierarchy.c, 47
- xml\_hierarchy\_header
  - xml\_hierarchy.c, 47
- xml\_hw\_component\_string\_convert
  - xml\_if.h, 59
- xml\_hw\_connection\_string\_convert
  - xml\_if.h, 59
- xml\_hw\_group\_string\_convert
  - xml\_if.h, 59
- xml\_hw\_instance\_string\_convert
  - xml\_if.h, 59
- xml\_if.c, 51
  - atoi\_null, 52
  - strcasecmp\_null, 53
  - xml\_add\_process\_argument, 53
  - xml\_add\_to\_hw\_group, 53
  - xml\_close\_declare, 53
  - xml\_declare\_function, 53
  - xml\_load\_exec\_component, 54
  - xml\_load\_exec\_instance, 54
  - xml\_load\_hw\_component, 54
  - xml\_load\_hw\_connection, 54
  - xml\_load\_hw\_group, 55
  - xml\_load\_hw\_instance, 55
  - xml\_load\_sw\_component, 55
  - xml\_load\_sw\_instance, 55
- xml\_if.h, 56
  - atoi\_null, 57
  - strcasecmp\_null, 57
  - xml\_add\_process\_argument, 58
  - xml\_add\_to\_hw\_group, 58
  - xml\_close\_declare, 58
  - xml\_connect\_hw\_connection, 58
  - xml\_connect\_hw\_group, 58
  - xml\_connect\_hw\_instance, 58
  - xml\_connect\_sw\_allocation, 59
  - xml\_connect\_sw\_instance, 59
  - xml\_declare\_function, 59
  - xml\_hw\_component\_string\_convert, 59
  - xml\_hw\_connection\_string\_convert, 59
  - xml\_hw\_group\_string\_convert, 59
  - xml\_hw\_instance\_string\_convert, 59
  - xml\_info\_level, 61
  - xml\_load\_exec\_component, 59
  - xml\_load\_exec\_instance, 59
  - xml\_load\_hw\_component, 59
  - xml\_load\_hw\_connection, 60

- xml\_load\_hw\_group, 60
- xml\_load\_hw\_instance, 60
- xml\_load\_sw\_component, 60
- xml\_load\_sw\_instance, 61
- xml\_sw\_allocation\_string\_convert, 61
- xml\_sw\_instance\_string\_convert, 61
- xml\_sw\_task\_string\_convert, 61
- xml\_increase\_path
  - xml\_hierarchy.c, 46
  - xml\_hierarchy.h, 50
- xml\_indexes
  - xml\_hierarchy.c, 48
- xml\_info\_level
  - xml\_if.h, 61
  - xml\_main.c, 81
- xml\_input.c, 61
  - add\_instance\_to\_hw\_group, 64
  - create\_hw\_component, 64
  - create\_hw\_connection, 64
  - create\_hw\_group, 64
  - create\_hw\_instance, 64
  - create\_sw\_allocation, 65
  - create\_sw\_component, 65
  - create\_sw\_instance, 65
  - create\_sw\_task, 65
  - get\_hw\_component\_by\_name, 65
  - get\_hw\_component\_by\_path, 66
  - get\_hw\_component\_header, 66
  - get\_hw\_connection\_header, 66
  - get\_hw\_group\_by\_name, 66
  - get\_hw\_group\_by\_path, 66
  - get\_hw\_group\_header, 67
  - get\_hw\_instance\_by\_name, 67
  - get\_hw\_instance\_by\_path, 67
  - get\_hw\_instance\_header, 67
  - get\_sw\_allocation\_header, 67
  - get\_sw\_component\_by\_name, 68
  - get\_sw\_component\_by\_path, 68
  - get\_sw\_component\_header, 68
  - get\_sw\_instance\_by\_name, 68
  - get\_sw\_instance\_by\_path, 68
  - get\_sw\_instance\_header, 69
  - get\_sw\_task\_by\_name, 69
  - get\_sw\_task\_by\_path, 69
  - get\_sw\_task\_header, 69
  - hw\_component\_header, 70
  - hw\_connection\_header, 70
  - hw\_group\_header, 70
  - hw\_instance\_header, 70
  - sw\_allocation\_header, 70
  - sw\_component\_header, 70
  - sw\_instance\_header, 70
  - sw\_task\_header, 70
  - xml\_convert\_path, 69
- xml\_input.h, 70
  - add\_instance\_to\_hw\_group, 72
  - create\_hw\_component, 72
  - create\_hw\_connection, 72
  - create\_hw\_group, 73
  - create\_hw\_instance, 73
  - create\_sw\_allocation, 73
  - create\_sw\_component, 73
  - create\_sw\_instance, 73
  - create\_sw\_task, 74
  - get\_hw\_component\_by\_name, 74
  - get\_hw\_component\_by\_path, 74
  - get\_hw\_component\_header, 74
  - get\_hw\_connection\_header, 74
  - get\_hw\_group\_by\_name, 75
  - get\_hw\_group\_by\_path, 75
  - get\_hw\_group\_header, 75
  - get\_hw\_instance\_by\_name, 75
  - get\_hw\_instance\_by\_path, 75
  - get\_hw\_instance\_header, 76
  - get\_sw\_allocation\_header, 76
  - get\_sw\_component\_by\_name, 76
  - get\_sw\_component\_by\_path, 76
  - get\_sw\_component\_header, 76
  - get\_sw\_instance\_by\_name, 77
  - get\_sw\_instance\_by\_path, 77
  - get\_sw\_instance\_header, 77
  - get\_sw\_task\_by\_name, 77
  - get\_sw\_task\_by\_path, 77
  - get\_sw\_task\_header, 78
- xml\_list.c, 78
  - xml\_list\_add\_element, 78
  - xml\_list\_last\_element, 78
  - xml\_list\_size, 79
- xml\_list.h, 79
  - xml\_list\_add\_element, 79
  - xml\_list\_last\_element, 79
  - xml\_list\_size, 80
- xml\_list\_add\_element
  - xml\_list.c, 78
  - xml\_list.h, 79
- xml\_list\_last\_element
  - xml\_list.c, 78
  - xml\_list.h, 79
- xml\_list\_size
  - xml\_list.c, 79
  - xml\_list.h, 80
- xml\_load\_configuration\_parameter
  - xml\_configuration\_file.c, 42
  - xml\_configuration\_file.h, 43
- xml\_load\_exec\_component
  - xml\_if.c, 54
  - xml\_if.h, 59
- xml\_load\_exec\_instance

- xml\_if.c, [54](#)
  - xml\_if.h, [59](#)
- xml\_load\_hw\_component
  - xml\_if.c, [54](#)
  - xml\_if.h, [59](#)
- xml\_load\_hw\_connection
  - xml\_if.c, [54](#)
  - xml\_if.h, [60](#)
- xml\_load\_hw\_group
  - xml\_if.c, [55](#)
  - xml\_if.h, [60](#)
- xml\_load\_hw\_instance
  - xml\_if.c, [55](#)
  - xml\_if.h, [60](#)
- xml\_load\_sw\_component
  - xml\_if.c, [55](#)
  - xml\_if.h, [60](#)
- xml\_load\_sw\_instance
  - xml\_if.c, [55](#)
  - xml\_if.h, [61](#)
- xml\_main.c, [80](#)
  - read\_xml\_file, [80](#)
  - xml\_has\_platform\_description, [81](#)
  - xml\_info\_level, [81](#)
- xml\_main.h, [81](#)
  - read\_xml\_file, [81](#)
- xml\_obtain\_metrics.cpp, [81](#)
  - calculate\_amat, [82](#)
  - calculate\_ec, [82](#)
  - calculate\_hr, [82](#)
  - calculate\_ic, [82](#)
  - calculate\_latency, [82](#)
  - calculate\_mips, [82](#)
  - calculate\_msc, [82](#)
  - GET\_INSTANCE\_PARAMETER, [82](#)
  - obtain\_xml\_metrics, [82](#)
  - xml\_get\_tlm\_power\_consumption, [82](#)
- xml\_obtain\_metrics.h, [83](#)
  - obtain\_xml\_metrics, [83](#)
- xml\_parameter\_info, [28](#)
  - name, [28](#)
  - value, [28](#)
- xml\_scope\_simulation\_info
  - uc\_load\_xml.cpp, [38](#)
  - uc\_load\_xml.h, [40](#)
- xml\_scope\_simulation\_info\_t, [28](#)
  - debug\_level, [28](#)
  - sim\_time\_unit, [28](#)
  - simulation\_time, [28](#)
  - warnings, [28](#)
- xml\_search\_path
  - xml\_hierarchy.c, [46](#)
  - xml\_hierarchy.h, [50](#)
- xml\_select\_implementation
  - xml\_hierarchy.c, [46](#)
  - xml\_hierarchy.h, [50](#)
- xml\_select\_implementation\_array
  - xml\_hierarchy.c, [48](#)
- xml\_simulation\_parameters
  - xml\_hierarchy.c, [47](#)
  - xml\_hierarchy.h, [50](#)
- xml\_string\_index\_convert
  - xml\_hierarchy.c, [47](#)
  - xml\_hierarchy.h, [51](#)
- xml\_sw\_allocation\_string\_convert
  - xml\_if.h, [61](#)
- xml\_sw\_instance\_string\_convert
  - xml\_if.h, [61](#)
- xml\_sw\_task\_string\_convert
  - xml\_if.h, [61](#)
- xml\_up\_hierarchy
  - xml\_hierarchy.c, [47](#)
  - xml\_hierarchy.h, [51](#)