CellML 1.0 C++ API Reference Manual

0.1

Generated by Doxygen 1.2.17

Sat Nov 20 23:13:44 2004

Contents

1 CellML 1.0 C++ API Hierarchical Index

1.1 CellML 1.0 C++ API Class Hierarchy

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

2 CellML 1.0 C++ API Compound Index

2.1 CellML 1.0 C++ API Compound List

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

3 CellML 1.0 C++ API Class Documentation

3.1 CellML_Component Class Reference

The `CellML_Component` is the main class used to define models.

Public Methods

Constructors and assignment operators.

- **CellML_Component** (const **CellML_Model** &parentModel)
  Constructor for components.
- **CellML_Component** (const CellML_Component &other)
  Copy constructor.
- CellML_Component & **operator=** (const CellML_Component &other)
  Assignment operator.

Destructor.

- ~**CellML_Component** ()
  Destructor for CellML_Component.

Equality.

- bool **equals** (const CellML_Component &other) const
  Check for equality of components.

Get functions.

- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this component object.
- const char * **getName** () const
  Gets the name of the component.
- const **CellML_UnitsList** & **getUnitsList** () const
  Get the units list from this component.
- **CellML_UnitsList** & **getUnitsList** ()
  Get the units list from this component.
- const **CellML_VariableList** & **getVariableList** () const
  Get the variable list from this component.
- **CellML_VariableList** & **getVariableList** ()
  Get the variable list from this component.
- const **CellML_MathMLDocumentList** * **getMathMLDocumentList** () const
  Get the list of MathML Math elements.
- **CellML_MathMLDocumentList** * **getMathMLDocumentList** ()
  Get the list of MathML Math elements.

Cloning function.

- CellML_Component **clone** (const bool deep) const
  Returns a duplicate of this component.

Query functions.

- bool **isValid** () const
  Test whether this component is valid.
- bool **isParentOf** (const CellML_Component &other) const
  Test if this component is a parent of another component.
- bool **isSiblingOf** (const CellML_Component &other) const
  Test if this component is a sibling of another component.
- bool **isEncapsulatedBy** (const CellML_Component &other) const
  Test if this component is encapsulated by another component.
- bool **isHiddenFrom** (const CellML_Component &other) const
  Test if this component is hidden from another component.

Set functions.

- void **setName** (const char *name)
  Sets the name of the component.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML Component into a DOM node.

Public Attributes

Factory methods.

- **CellML_Units createUnits** () const
  Create a units.
- **CellML_Variable createVariable** () const
  Create a variable.

Private Methods

- **CellML_Component** ()
  Default constructor for CellML_Component.
- void **setParentModel** (const **CellML_Model** &model)
  Sets the parent model of this component.

Private Attributes

- ComponentContent * **fContent**

  The content of the component.

- const **CellML_Model** * **fParentModel**

  The parent of this component.

Friends

- class **CellML_ComponentList**

3.1.1 Detailed Description

The `CellML_Component` is the main class used to define models.

Components hold the variable declarations, mathematical equations, and reaction definitions.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 CellML_Component::CellML_Component (const CellML_Model & *parentModel*)

Constructor for components.

3.1.2.2 CellML_Component::CellML_Component (const CellML_Component & *other*)

Copy constructor.

The copy constructor will return a new `CellML_Component` object that has the same content object as the object being copied. If you want to have a new content object you need to use the `clone` method.

Parameters:
other The object to be copied.

See also:
clone() (p. 8).

3.1.2.3 CellML_Component::~CellML_Component ()

Destructor for CellML_Component.

3.1.2.4 CellML_Component::CellML_Component () [private]

Default constructor for CellML_Component.

3.1.3 Member Function Documentation

3.1.3.1 CellML_Component CellML_Component::clone (const bool *deep*) const

Returns a duplicate of this component.

This function serves as a generic copy constructor for components.

Parameters:

deep If `true`, recursively clone the child objects of the component (i.e. variables, equations,...); if `false`, clone only the component itself.

Returns :

The duplicate component.

3.1.3.2 bool CellML_Component::equals (const CellML_Component & *other*) const

Check for equality of components.

This will recursively descend through this component and the other component testing for equality.

Parameters:

other The component reference with which `this` object is compared.

Returns :

True if both components are identical; return false otherwise.

3.1.3.3 void CellML_Component::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a component from the given DOM source node which should be a `component` element in the CellML namespace.

Parameters:

srcNode The DOM node containing a CellML component description.

Exceptions:

CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a component object from a DOM source.

DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.1.3.4 CellML_MathMLDocumentList* CellML_Component::getMathMLDocumentList ()

Get the list of MathML Math elements.

This is a special case (??) because we always store mathematics in MathML so for now simply provide direct access to the math element list so people can then access the MathML DOM methods directly. This is the easiest way to let applications create/modify equations themselves - for now ? **Need to add mor methods for accessing and modifying and creating equations!!**

Returns :

The list of `MathMLDocument`s for this component.

3.1.3.5 const CellML_MathMLDocumentList* CellML_Component::getMathMLDocumentList () const

Get the list of MathML Math elements.

This is a special case (??) because we always store mathematics in MathML so for now simply provide direct access to the math element list so people can then access the MathML DOM methods directly. This is the easiest way to let applications create/modify equations themselves - for now ? **Need to add mor methods for accessing and modifying and creating equations!!**

Returns :
The list of `MathMLDocument`s for this component.

3.1.3.6 const char* CellML_Component::getName () const

Gets the name of the component.

Returns :
A null-terminated string representation of the name of `this` component.

3.1.3.7 const CellML_Model& CellML_Component::getParentModel () const

Gets the parent model of this component object.

Returns :
A constant reference to the parent CellML Model of this component object.

3.1.3.8 CellML_UnitsList& CellML_Component::getUnitsList ()

Get the units list from this component.

Unlike a **CellML_Model** (p. 118)'s units list, a component's units list is not initialised with the standard units.

Returns :
A reference to this model's units list.

3.1.3.9 const CellML_UnitsList& CellML_Component::getUnitsList () const

Get the units list from this component.

Unlike a **CellML_Model** (p. 118)'s units list, a component's units list is not initialised with the standard units.

Returns :
A constant reference to this component's units list.

3.1.3.10 CellML_VariableList& CellML_Component::getVariableList ()

Get the variable list from this component.

Returns :
A reference to this model's variable list.

3.1.3.11 const CellML_VariableList& CellML_Component::getVariableList () const

Get the variable list from this component.

Returns :
A constant reference to this component's variable list.

3.1.3.12 bool CellML_Component::isEncapsulatedBy (const CellML_Component & *other*) const

Test if this component is encapsulated by another component.

The rules for mapping a variable to other variables depend on the encapsulation of the component that owns the variable. This hierarchy divides the rest of the components in the model into *parent*, *sibling*, *encapsulated*, and *hidden* sets (see Section 3.2.3 of the CellML 1.0 Specification). This method returns `true` if `this` component is encapsulated by the `other` component. If this method is called before any encapsulation hierarchy is defined, then the result will always be `false`.

Parameters:
other The component with which to test for a encapsulation relationship.

Returns :
true if this component is encapsulated by the `other` component; false otherwise.

3.1.3.13 bool CellML_Component::isHiddenFrom (const CellML_Component & *other*) const

Test if this component is hidden from another component.

The rules for mapping a variable to other variables depend on the encapsulation of the component that owns the variable. This hierarchy divides the rest of the components in the model into *parent*, *sibling*, *encapsulated*, and *hidden* sets (see Section 3.2.3 of the CellML 1.0 Specification). This method returns `true` if `this` component is hidden from the `other` component. If this method is called before any encapsulation hierarchy is defined, then the result will always be `false`.

Parameters:
other The component with which to test for a hidden relationship.

Returns :
true if this component is hidden from the `other` component; false otherwise.

3.1.3.14 bool CellML_Component::isParentOf (const CellML_Component & *other*) const

Test if this component is a parent of another component.

The rules for mapping a variable to other variables depend on the encapsulation of the component that owns the variable. This hierarchy divides the rest of the components in the model into *parent*, *sibling*, *encapsulated*, and *hidden* sets (see Section 3.2.3 of the CellML 1.0 Specification). This method returns `true` if `this` component is a parent component of the `other` component. If this method is called before any encapsulation hierarchy is defined, then the result will always be `false`.

Parameters:
other The component with which to test for a parent relationship.

Returns :
true if this component is a parent of the `other` component; false otherwise.

3.1.3.15 bool CellML_Component::isSiblingOf (const CellML_Component & *other*) const

Test if this component is a sibling of another component.

The rules for mapping a variable to other variables depend on the encapsulation of the component that owns the variable. This hierarchy divides the rest of the components in the model into *parent*, *sibling*, *encapsulated*, and *hidden* sets (see Section 3.2.3 of the CellML 1.0 Specification). This method returns `true` if `this` component is a sibling component of the `other` component. If this method is called before any encapsulation hierarchy is defined, then the result will always be `true`.

Parameters:
other The component with which to test for a sibling relationship.

Returns :
true if this component is a sibling of the `other` component; false otherwise.

3.1.3.16 bool CellML_Component::isValid () const

Test whether this component is valid.

What does this mean for a CellML Component ?? This should only be called once the component has been fully populated and the calling routine wants to check the validity of the current contents of the component.

Returns :
True if the component is valid; false otherwise.

3.1.3.17 CellML_Component& CellML_Component::operator= (const CellML_Component & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the `clone` method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 8).

3.1.3.18 void CellML_Component::setName (const char * *name*)

Sets the name of the component.

Takes a copy of the `name` so it can be safely freed by the calling routine.

Parameters:
name The name to be assigned to this component.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the `name` is an illegal CellML identifier.

3.1.3.19 void CellML_Component::setParentModel (const CellML_Model & *model*) [private]

Sets the parent model of this component.

Parameters:
model The parent model of this component.

3.1.3.20 DOMNode* CellML_Component::toNode () const

Translate this CellML Component into a DOM node.

Returns :
The DOM Node created which represents the current contents of this component object.

3.1.4 Member Data Documentation

3.1.4.1 CellML_Units CellML_Component::createUnits() const

Create a units.

Creates an empty **CellML_Units** (p. 149) object as a child of this component.

Returns :
The units object created.

3.1.4.2 CellML_Variable CellML_Component::createVariable() const

Create a variable.

Creates an empty **CellML_Variable** (p. 163) object as a child of this component.

Returns :
The variable object created.

3.1.4.3 ComponentContent* CellML_Component::fContent [private]

The content of the component.

3.1.4.4 const CellML_Model* CellML_Component::fParentModel [private]

The parent of this component.

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

- CellML_Component.hpp

3.2 CellML_ComponentList Class Reference

A class for representing lists of **CellML_Component** (p. 4).

Public Methods

Constructors and assignment operators.

- **CellML_ComponentList** (const **CellML_Model** &parentModel)
  Constructor for lists belonging to models.
- **CellML_ComponentList** (const CellML_ComponentList &other)
  Copy constructor.
- CellML_ComponentList & **operator=** (const CellML_ComponentList &other)
  Assignment operator.

Destructor.

- **~CellML_ComponentList** ()
  Destructor for CellML_ComponentList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_ComponentList &other) const
  Test two lists are equal.
- const **CellML_Component** & **get** (const int index) const
  Get a component from the list.
- const **CellML_Component** & **get** (const char *name) const
  Get a component from the list.
- void **append** (**CellML_Component** &newComponent)
  Append a component to the list.
- void **append** (const CellML_ComponentList &other)
  Append a list of components to this list.

- **CellML_Component remove** (const **CellML_Component** &oldComponent)
  Remove the given component from the list.
- **CellML_Component replace** (**CellML_Component** &newComponent, const **CellML_Component** &oldComponent)
  Replace a component object with another.
- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this list object.

Private Methods

- **CellML_ComponentList** ()
  Default constructor.

3.2.1 Detailed Description

A class for representing lists of **CellML_Component** (p. 4).

Enforces the CellML requirement for unique names within the list.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 CellML_ComponentList::CellML_ComponentList (const CellML_Model & *parentModel*)

Constructor for lists belonging to models.

Parameters:
parentModel The parent of this list.

3.2.2.2 CellML_ComponentList::CellML_ComponentList (const CellML_ComponentList & *other*)

Copy constructor.

Will create a new list and populate it with clones of the components in the source list.

Parameters:
other The list to be copied.

See also:
CellML_Component::CellML_Component(const CellML_Component&) (p. 7) , **CellML_Component::clone**() (p. 8) , **append**(const CellML_ComponentList&) (p. 15)

3.2.2.3 CellML_ComponentList::~CellML_ComponentList ()

Destructor for CellML_ComponentList.

3.2.2.4 CellML_ComponentList::CellML_ComponentList () [private]

Default constructor.

Will construct an empty list.

3.2.3 Member Function Documentation

3.2.3.1 void CellML_ComponentList::append (const CellML_ComponentList & *other*)

Append a list of components to this list.

Appends clones of the components in the source list to this list, ensuring that the components added to the list have the same parent as this list.

Parameters:
other The list of components to add.

Exceptions:
CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if any components in the `other` list have the same name as a component already in the list.

See also:
CellML_Component::clone() (p. 8).

3.2.3.2 void CellML_ComponentList::append (CellML_Component & *newComponent*)

Append a component to the list.

Adds `newComponent` to the end of the list. Appending the component to the list will ensure that `newComponent` will have the same parent as this list.

Parameters:
newComponent The component object to add to the end of this list.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if `newComponent` has an invalid name.
CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if a component with the same name as `newComponent` is already in the list.

3.2.3.3 bool CellML_ComponentList::equals (const CellML_ComponentList & *other*) const

Test two lists are equal.

Test the contents of `this` and `other` lists for equality. The list are deemed equal if they are both the same length and for each entry in `this` list there is an equal entry in the `other` list.

Parameters:
other The list to compare `this` list to.

Returns :
true if `this` list is equal to `other`; false otherwise.

3.2.3.4 const CellML_Component& CellML_ComponentList::get (const char * *name*) const

Get a component from the list.

Used to get a component object from this list by its name.

Parameters:
name The name of the desired component.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `name` is not a name of one of the components currently in this component list.

3.2.3.5 const CellML_Component& CellML_ComponentList::get (const int *index*) const

Get a component from the list.

Used to get a component object from this list by its position in the list.

Parameters:
index The index of the desired component (valid range is from 0 to **length**() (p. 16)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is outside the valid range for this component list.

See also:
length() (p. 16)

3.2.3.6 const CellML_Model& CellML_ComponentList::getParentModel () const

Gets the parent model of this list object.

Returns :
A constant reference to the parent CellML Model of this list object.

3.2.3.7 bool CellML_ComponentList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.2.3.8 int CellML_ComponentList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.2.3.9 CellML_ComponentList& CellML_ComponentList::operator= (const CellML_ComponentList & *other*)

Assignment operator.

Will assign a copy of the source list to the destination list.

Parameters:
other The source to be assigned.

See also:
CellML_ComponentList(const CellML_ComponentList&) (p. 14)

3.2.3.10 CellML_Component CellML_ComponentList::remove (const CellML_Component & *oldComponent*)

Remove the given component from the list.

Removes `oldComponent` from the list and returns it.

Parameters:
oldComponent The component object to remove from the list.

Returns :
The component removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldComponent` is not found in this list.

3.2.3.11 CellML_Component CellML_ComponentList::replace (CellML_Component & *newComponent*, const CellML_Component & *oldComponent*)

Replace a component object with another.

Replaces `oldComponent` with `newComponent`. `newComponent` will be put into the same position in the list that is vacated by `oldComponent` - other than that this method is essentially the same as doing a remove(oldComponent) followed by a append(newComponent). This method will also ensure that `newComponent` has the same parent as this list.

Parameters:
newComponent The component to add to the list.

oldComponent The component to remove from the list.

Returns :
The component removed from the list.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if `newComponent` has an invalid name. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if a component with the same name as `newComponent` is already in the list (after `oldComponent` is removed from the list). If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldComponent` is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

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

- CellML_ComponentList.hpp

3.3 CellML_ComponentRef Class Reference

A `CellML_ComponentRef` defines a reference to a component from within a group description.

Public Methods

Constructors and assignment operators.

- **CellML_ComponentRef** (const **CellML_Group** &parentGroup)
  Constructor for CellML_ComponentRef.
- **CellML_ComponentRef** (const **CellML_Group** &parentGroup, const CellML_ComponentRef &parentComponentRef)
  Constructor for CellML_ComponentRef.
- **CellML_ComponentRef** (const CellML_ComponentRef &other)
  Copy constructor.
- CellML_ComponentRef & **operator=** (const CellML_ComponentRef &other)
  Assignment operator.

Destructor.

- ~**CellML_ComponentRef** ()
  Destructor for CellML_ComponentRef.

Equality.

- bool **equals** (const CellML_ComponentRef &other) const
  Check for equality of component_ref's.

Get functions.

- const **CellML_Group** & **getParentGroup** () const
  Gets the parent group of this component_ref object.
- const CellML_ComponentRef & **getParentComponentRef** () const
  Gets the parent component_ref of this component_ref object.
- const char * **getComponentName** () const
  Gets the name of the component referenced.

- const **CellML_Component** & **getComponent** () const
  Gets a reference to the component referenced.
- const **CellML_ComponentRefList** & **getComponentRefList** () const
  Get the component_ref list from this component_ref.
- **CellML_ComponentRefList** & **getComponentRefList** ()
  Get the component_ref list from this component_ref.

Cloning function.

- CellML_ComponentRef **clone** (const bool deep) const
  Returns a duplicate of this component_ref object.

Query functions.

- bool **isValid** () const
  Test whether this component_ref is valid.
- bool **hasParentComponentRef** () const
  Test if this component_ref has a component_ref parent.

Set functions.

- void **setComponentName** (const char *name)
  Sets the name of the component referenced.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML ComponentRef into a DOM node.

Public Attributes

- CellML_ComponentRef **createComponentRef** () const
  Create a component_ref as a child of this component_ref.

Private Methods

- **CellML_ComponentRef** ()
  Default constructor for CellML_ComponentRef.
- void **setParentGroup** (const **CellML_Group** &group)

Sets the parent group of this component_ref object.

- void **setParentComponentRef** (const CellML_ComponentRef &componentRef)

  Sets the parent component_ref of this component_ref object.

Private Attributes

- const **CellML_Group** * **fParentGroup**

  The parent group.

- const CellML_ComponentRef * **fParentComponentRef**

  The parent component_ref (if there is one).

- ComponentRefContent * **fContent**

  The content of the component_ref.

Friends

- class **CellML_ComponentRefList**

3.3.1 Detailed Description

A `CellML_ComponentRef` defines a reference to a component from within a group description.

`CellML_ComponentRef` objects are used to specify hierarchies of components and user defined model structure.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 CellML_ComponentRef::CellML_ComponentRef (const CellML_Group & *parentGroup*)

Constructor for CellML_ComponentRef.

3.3.2.2 CellML_ComponentRef::CellML_ComponentRef (const CellML_Group & *parentGroup*, const CellML_ComponentRef & *parentComponentRef*)

Constructor for CellML_ComponentRef.

3.3.2.3 CellML_ComponentRef::CellML_ComponentRef (const CellML_ComponentRef & *other*)

Copy constructor.

The copy constructor will return a new `CellML_ComponentRef` that has the same content object as the object being copied. If you want to have a new content object you need to use the `clone` method.

Parameters:

other The object to be copied.

See also:
clone() (p. 21).

3.3.2.4 CellML_ComponentRef::~CellML_ComponentRef ()

Destructor for CellML_ComponentRef.

3.3.2.5 CellML_ComponentRef::CellML_ComponentRef () [private]

Default constructor for CellML_ComponentRef.

3.3.3 Member Function Documentation

3.3.3.1 CellML_ComponentRef CellML_ComponentRef::clone (const bool *deep*) const

Returns a duplicate of this component_ref object.

This function serves as a generic copy constructor for component_refs.

Parameters:
deep If `true`, recursively clone the child objects of the component_ref; if `false`, clone only the component_ref itself.

Returns :
The duplicate component_ref.

3.3.3.2 bool CellML_ComponentRef::equals (const CellML_ComponentRef & *other*) const

Check for equality of component_ref's.

This will test for equality between `this` component_ref and `other`.

Parameters:
other The component_ref reference with which `this` object is compared.

Returns :
True if both component_refs are identical; return false otherwise.

3.3.3.3 void CellML_ComponentRef::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a component_ref from the given DOM source node which should be a `component_ref` element in the CellML namespace.

Parameters:
srcNode The DOM node containing a CellML component_ref description.

Exceptions:
CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a component_ref object from a DOM source.
DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.3.3.4 const CellML_Component& CellML_ComponentRef::getComponent () const

Gets a reference to the component referenced.

Convenience method to get a reference to the actual component referenced by this component_ref.

Returns :
A constant reference to the component referenced.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the component name is currently not set or is invalid.

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if the component referenced by this component_ref can not be found in the current model.

3.3.3.5 const char* CellML_ComponentRef::getComponentName () const

Gets the name of the component referenced.

Returns :
A null-terminated string representation of the name of the component referenced.

3.3.3.6 CellML_ComponentRefList& CellML_ComponentRef::getComponentRefList ()

Get the component_ref list from this component_ref.

Returns :
A reference to this component_ref's component_ref list.

3.3.3.7 const CellML_ComponentRefList& CellML_ComponentRef::getComponentRefList () const

Get the component_ref list from this component_ref.

Returns :
A constant reference to this component_ref's component_ref list.

3.3.3.8 const CellML_ComponentRef& CellML_ComponentRef::getParentComponentRef () const

Gets the parent component_ref of this component_ref object.

Returns :
A constant reference to the parent CellML ComponentRef of this component_ref object.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if this is a top-level component_ref - i.e., it has no parent component_ref.

3.3.3.9 const CellML_Group& CellML_ComponentRef::getParentGroup () const

Gets the parent group of this component_ref object.

Returns :
A constant reference to the parent CellML Group of this component_ref object.

3.3.3.10 bool CellML_ComponentRef::hasParentComponentRef () const

Test if this component_ref has a component_ref parent.

Convenience method to determine if this component_ref element is the child of a component_ref element.

Returns :
True if this component_ref has a parent component_ref; false otherwise.

3.3.3.11 bool CellML_ComponentRef::isValid () const

Test whether this component_ref is valid.

What does this mean for a CellML ComponentRef ?? This should only be called once the component_ref has been fully populated and the calling routine wants to check the validity of the current contents of the component_ref.

Returns :
True if the component_ref is valid; false otherwise.

3.3.3.12 CellML_ComponentRef& CellML_ComponentRef::operator= (const CellML_ComponentRef & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the `clone` method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 21).

3.3.3.13 void CellML_ComponentRef::setComponentName (const char * *name*)

Sets the name of the component referenced.

Takes a copy of the `name` so it can be safely freed by the calling routine.

Parameters:
name The name of the component to be referenced.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the `name` is an illegal CellML identifier.
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if the `name` is null.

3.3.3.14 void CellML_ComponentRef::setParentComponentRef (const CellML_ComponentRef & *componentRef*) [private]

Sets the parent component_ref of this component_ref object.

3.3.3.15 void CellML_ComponentRef::setParentGroup (const CellML_Group & *group*) [private]

Sets the parent group of this component_ref object.

3.3.3.16 DOMNode* CellML_ComponentRef::toNode () const

Translate this CellML ComponentRef into a DOM node.

Returns :
The DOM Node created which represents the current contents of this component_ref object.

3.3.4 Member Data Documentation

3.3.4.1 CellML_ComponentRef CellML_ComponentRef::createComponentRef() const

Create a component_ref as a child of this component_ref.

Returns :
The new child component_ref.

3.3.4.2 ComponentRefContent* CellML_ComponentRef::fContent [private]

The content of the component_ref.

3.3.4.3 const CellML_ComponentRef* CellML_ComponentRef::fParentComponentRef [private]

The parent component_ref (if there is one).

3.3.4.4 const CellML_Group* CellML_ComponentRef::fParentGroup [private]

The parent group.

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

- CellML_ComponentRef.hpp

3.4 CellML_ComponentRefList Class Reference

A class for representing lists of **CellML_ComponentRef** (p. 18)s.

Public Methods

Constructors and assignment operators.

- **CellML_ComponentRefList** (const **CellML_Group** &parentGroup)
  Constructor for lists belonging to groups.
- **CellML_ComponentRefList** (const **CellML_ComponentRef** &parentComponentRef)
  Constructor for lists belonging to component_refs.
- **CellML_ComponentRefList** (const CellML_ComponentRefList &other)
  Copy constructor.
- CellML_ComponentRefList & **operator=** (const CellML_ComponentRefList &other)
  Assignment operator.

Destructor.

- **~CellML_ComponentRefList** ()
  Destructor for CellML_ComponentRefList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_ComponentRefList &other) const
  Test two lists are equal.
- const **CellML_ComponentRef** & **get** (const int index) const
  Get a component_ref from the list.
- void **append** (**CellML_ComponentRef** &newComponentRef)
  Append a component_ref to the list.
- void **append** (const CellML_ComponentRefList &other)
  Append a list of component_ref's to this list.
- **CellML_ComponentRef remove** (const **CellML_ComponentRef** &oldComponentRef)
  Remove the given component_ref from the list.
- **CellML_ComponentRef replace** (**CellML_ComponentRef** &newComponentRef, const **CellML_ComponentRef** &oldComponentRef)
  Replace a component_ref object with another.
- const **CellML_Group** & **getParentGroup** () const
  Gets the parent group of this list object.
- const **CellML_ComponentRef** & **getParentComponentRef** () const
  Gets the parent component_ref of this component_ref list.

Private Methods

- **CellML_ComponentRefList** ()
  Default constructor.

3.4.1 Detailed Description

A class for representing lists of **CellML_ComponentRef** (p. 18)s.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 CellML_ComponentRefList::CellML_ComponentRefList (const CellML_Group & *parentGroup*)

Constructor for lists belonging to groups.

Parameters:
parentGroup The parent of this list.

3.4.2.2 CellML_ComponentRefList::CellML_ComponentRefList (const CellML_ComponentRef & *parentComponentRef*)

Constructor for lists belonging to component_refs.

Parameters:
parentComponentRef The parent of this list.

3.4.2.3 CellML_ComponentRefList::CellML_ComponentRefList (const CellML_ComponentRefList & *other*)

Copy constructor.

Will create a new list and populate it with clones of the components in the source list.

Parameters:
other The list to be copied.

See also:
CellML_ComponentRef::CellML_ComponentRef(const CellML_ComponentRef&) (p. 20) , **CellML_ComponentRef::clone**() (p. 21) , **append**(const CellML_ComponentRefList&) (p. 27)

3.4.2.4 CellML_ComponentRefList::~CellML_ComponentRefList ()

Destructor for CellML_ComponentRefList.

3.4.2.5 CellML_ComponentRefList::CellML_ComponentRefList () [private]

Default constructor.

Will construct an empty list.

3.4.3 Member Function Documentation

3.4.3.1 void CellML_ComponentRefList::append (const CellML_ComponentRefList & *other*)

Append a list of component_ref's to this list.

Appends clones of the component_refs in the source list to this list, ensuring that the component_refs added to the list have the same parent as this list.

Parameters:
other The list of component_ref's to add.

See also:
CellML_ComponentRef::clone() (p. 21).

3.4.3.2 void CellML_ComponentRefList::append (CellML_ComponentRef & *newComponentRef*)

Append a component_ref to the list.

Adds `newComponentRef` to the end of the list. Appending the component_ref to the list will ensure that `newComponentRef` will have the same parent as this list.

Parameters:
newComponentRef The component_ref object to add to the end of this list.

3.4.3.3 bool CellML_ComponentRefList::equals (const CellML_ComponentRefList & *other*) const

Test two lists are equal.

Test the contents of `this` and `other` lists for equality. The list are deemed equal if they are both the same length and for each entry in `this` list there is an equal entry in the `other` list.

Parameters:
other The list to compare `this` list to.

Returns :
true if `this` list is equal to `other`; false otherwise.

3.4.3.4 const CellML_ComponentRef& CellML_ComponentRefList::get (const int *index*) const

Get a component_ref from the list.

Used to get a component_ref object from this list by its position in the list.

Parameters:
index The index of the desired component_ref (valid range is from 0 to **length**() (p. 28)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is outside the valid range for this component_ref list.

See also:
length() (p. 28)

3.4.3.5 const CellML_ComponentRef& CellML_ComponentRefList::getParentComponentRef () const

Gets the parent component_ref of this component_ref list.

Returns :
A constant reference to the parent CellML ComponentRef of this component_ref list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if this is a top-level component_ref list - i.e., it is a direct child of a CellML group

3.4.3.6 const CellML_Group& CellML_ComponentRefList::getParentGroup () const

Gets the parent group of this list object.

Returns :
A constant reference to the parent CellML Group of this list object.

3.4.3.7 bool CellML_ComponentRefList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.4.3.8 int CellML_ComponentRefList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.4.3.9 CellML_ComponentRefList& CellML_ComponentRefList::operator= (const CellML_ComponentRefList & *other*)

Assignment operator.

Will assign a copy of the source list to the destination list.

Parameters:
other The source to be assigned.

See also:
CellML_ComponentRefList(const CellML_ComponentRefList&) (p. 26)

3.4.3.10 CellML_ComponentRef CellML_ComponentRefList::remove (const CellML_ComponentRef & *oldComponentRef*)

Remove the given component_ref from the list.

Removes `oldComponentRef` from the list and returns it.

Parameters:
oldComponentRef The component_ref object to remove from the list.

Returns :
The component_ref removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldComponentRef` is not found in this list.

3.4.3.11 CellML_ComponentRef CellML_ComponentRefList::replace (CellML_ComponentRef & *newComponentRef*, const CellML_ComponentRef & *oldComponentRef*)

Replace a component_ref object with another.

Replaces `oldComponentRef` with `newComponentRef`. `newComponentRef` will be put into the same position in the list that is vacated by `oldComponentRef` - other than that this method is essentially the same as doing a remove(oldComponentRef) followed by a append(newComponentRef). This method will also ensure that `newComponentRef` has the same parent as this list.

Parameters:
newComponentRef The component_ref to add to the list.

oldComponentRef The component_ref to remove from the list.

Returns :
The component_ref removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldComponentRef` is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

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

- CellML_ComponentRefList.hpp

3.5 CellML_Connection Class Reference

`CellML_Connection`s are used to connect variables together, allowing variables to be mapped from one component to another.

Public Methods

Constructors and assignment operators.

- **CellML_Connection** (const **CellML_Model** &parentModel)
  Constructor for connections.
- **CellML_Connection** (const CellML_Connection &other)
  Copy constructor.
- CellML_Connection & **operator=** (const CellML_Connection &other)
  Assignment operator.

Destructor.

- **~CellML_Connection** ()
  Destructor for CellML_Connection.

Equality.

- bool **equals** (const CellML_Connection &other) const
  Check for equality of connections.

Get functions.

- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this connection object.
- const **CellML_MapVariablesList** & **getMapVariablesList** () const
  Get the map_variables' list from this connection.
- **CellML_MapVariablesList** & **getMapVariablesList** ()
  Get the map_variables' list from this connection.
- const **CellML_MapComponents** & **getMapComponents** () const
  Get the map_components for this connection.

Cloning function.

- CellML_Connection **clone** (const bool deep) const
  Returns a duplicate of this connection.

Query functions.

- bool **isValid** () const
  Test whether this connection is valid.

Set functions.

- void **setMapComponents** (const **CellML_MapComponents** &mapComponents)
  Sets the map_components for this connection.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML Connection into a DOM node.

Static Public Methods

Other functions.

- bool **validateConnection** (const **CellML_Variable** &inV, const **CellML_Component** &inC, const **CellML_Variable** &outV, const **CellML_Component** &outC)
  Check the connection between two variables.

Public Attributes

Factory methods.

- **CellML_MapVariables createMapVariables** () const
  Create a map_variables.
- **CellML_MapComponents createMapComponents** () const
  Create a map_components.

Private Methods

- **CellML_Connection** ()
  Default constructor for CellML_Connection.
- void **setParentModel** (const **CellML_Model** &model)
  Sets the parent model of this connection.

Private Attributes

- const **CellML_Model** * **fParentModel**
  The parent of the connection.
- ConnectionContent * **fContent**
  The content of the connection.

Friends

- class **CellML_ConnectionList**

3.5.1 Detailed Description

`CellML_Connection`s are used to connect variables together, allowing variables to be mapped from one component to another.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 CellML_Connection::CellML_Connection (const CellML_Model & *parentModel*)

Constructor for connections.

3.5.2.2 CellML_Connection::CellML_Connection (const CellML_Connection & *other*)

Copy constructor.

The copy constructor will return a new `CellML_Connection` object that has the same content object as the object being copied. If you want to have a new content object you need to use the `clone` method.

Parameters:
other The object to be copied.

See also:
clone() (p. 32).

3.5.2.3 CellML_Connection::~CellML_Connection ()

Destructor for CellML_Connection.

3.5.2.4 CellML_Connection::CellML_Connection () `[private]`

Default constructor for CellML_Connection.

3.5.3 Member Function Documentation

3.5.3.1 CellML_Connection CellML_Connection::clone (const bool *deep*) const

Returns a duplicate of this connection.

This function serves as a generic copy constructor for connections. Cloning a connection results in a new content object being created and initialised with the contents of the connection being cloned.

Parameters:
deep If `true`, recursively clone the child objects of the connection; if `false`, clone only the connection itself.

Returns :
The duplicate connection.

3.5.3.2 bool CellML_Connection::equals (const CellML_Connection & *other*) const

Check for equality of connections.

This will recursively descend through this connection and the other connection testing for equality.

Parameters:
other The connection reference with which `this` object is compared.

Returns :
True if both connections are identical; return false otherwise.

3.5.3.3 void CellML_Connection::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a connection from the given DOM source node which should be a `connection` element in the CellML namespace.

Parameters:
srcNode The DOM node containing a CellML connection description.

Exceptions:
CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a connection object from a DOM source.
DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.5.3.4 const CellML_MapComponents& CellML_Connection::getMapComponents () const

Get the map_components for this connection.

Returns :
A constant reference to the map_components child of this connection.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if the map_components for this connection has not been set.

3.5.3.5 CellML_MapVariablesList& CellML_Connection::getMapVariablesList ()

Get the map_variables' list from this connection.

Returns :
A reference to this connection's MapVariables' list.

3.5.3.6 const CellML_MapVariablesList& CellML_Connection::getMapVariablesList () const

Get the map_variables' list from this connection.

Returns :
A const reference to this connection's MapVariables' list.

3.5.3.7 const CellML_Model& CellML_Connection::getParentModel () const

Gets the parent model of this connection object.

Returns :
A constant reference to the parent CellML Connection of this group object.

3.5.3.8 bool CellML_Connection::isValid () const

Test whether this connection is valid.

What does this mean for a CellML Connection ?? This should only be called once the connection has been fully populated and the calling routine wants to check the validity of the current contents of the connection.

Returns :
True if the connection is valid; false otherwise.

3.5.3.9 CellML_Connection& CellML_Connection::operator= (const CellML_Connection & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the `clone` method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 32).

3.5.3.10 void CellML_Connection::setMapComponents (const CellML_MapComponents & *mapComponents*)

Sets the map_components for this connection.

If `mapComponents` is successfully added to the connection its parent connection will be set to `this` connection.

Parameters:
mapComponents The map_components of this connection.

3.5.3.11 void CellML_Connection::setParentModel (const CellML_Model & *model*) [private]

Sets the parent model of this connection.

Parameters:
model The parent model of this connection.

3.5.3.12 DOMNode* CellML_Connection::toNode () const

Translate this CellML Connection into a DOM node.

Returns :
The DOM Node created which represents the current contents of this connection object.

3.5.3.13 bool CellML_Connection::validateConnection (const CellML_Variable & *inV*, const CellML_Component & *inC*, const CellML_Variable & *outV*, const CellML_Component & *outC*) [static]

Check the connection between two variables.

Takes two component/variable pairs and determines if it is valid for them to be connected. Checks using any defined encapsulation hierarchies in the parent model.

Parameters:
inV The variable which should have either a public or private interface of "in".

inC The component parent of `inV`

outV The variable which should have either a public or private interface of "out".

outC The component parent of `outV`

Returns :
True if these variables can be connected; false otherwise.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `inV` does not have a public or private interface of "in".

3.5.4 Member Data Documentation

3.5.4.1 CellML_MapComponents CellML_Connection::createMapComponents() const

Create a map_components.

Creates an empty **CellML_MapComponents** (p. 54) object as a child of this connection.

Returns :
The map_components object created.

3.5.4.2 CellML_MapVariables CellML_Connection::createMapVariables() const

Create a map_variables.

Creates an empty **CellML_MapVariables** (p. 60) object as a child of this connection.

Returns :
The map_variables object created.

3.5.4.3 ConnectionContent* CellML_Connection::fContent [private]

The content of the connection.

3.5.4.4 const CellML_Model* CellML_Connection::fParentModel [private]

The parent of the connection.

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

- CellML_Connection.hpp

3.6 CellML_ConnectionList Class Reference

A class for representing lists of **CellML_Connection** (p. 29)s.

Public Methods

Constructors and assignment operators.

- **CellML_ConnectionList** (const **CellML_Model** &parentModel)
  Constructor for lists belonging to models.
- **CellML_ConnectionList** (const CellML_ConnectionList &other)
  Copy constructor.
- CellML_ConnectionList & **operator=** (const CellML_ConnectionList &other)
  Assignment operator.

Destructor.

- ~**CellML_ConnectionList** ()
  Destructor for CellML_ConnectionList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_ConnectionList &other) const
  Test two lists are equal.
- const **CellML_Connection** & **get** (const int index) const
  Get a connection from the list.
- void **append** (**CellML_Connection** &newConnection)
  Append a connection to the list.
- void **append** (const CellML_ConnectionList &other)
  Append a list of connection's to this list.
- **CellML_Connection remove** (const **CellML_Connection** &oldConnection)
  Remove the given connection from the list.

- **CellML_Connection replace** (**CellML_Connection** &newConnection, const **CellML_Connection** &oldConnection)
  Replace a connection object with another.
- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this list object.

Private Methods

- **CellML_ConnectionList** ()
  Default constructor.

3.6.1 Detailed Description

A class for representing lists of **CellML_Connection** (p. 29)s.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 CellML_ConnectionList::CellML_ConnectionList (const CellML_Model & *parentModel*)

Constructor for lists belonging to models.

Parameters:
parentModel The parent of this list.

3.6.2.2 CellML_ConnectionList::CellML_ConnectionList (const CellML_ConnectionList & *other*)

Copy constructor.

Copies only references to objects in the list, not the actual objects. Note that this does not change the parent of the list or the elements of the list. If you wish to create a copy of a list to initialise a new object you should first create an empty list and then append the existing list to the new list.

Parameters:
other The list to be copied.

See also:
append(const CellML_ConnectionList&) (p. 38)

3.6.2.3 CellML_ConnectionList::~CellML_ConnectionList ()

Destructor for CellML_ConnectionList.

3.6.2.4 CellML_ConnectionList::CellML_ConnectionList () [private]

Default constructor.

Will construct an empty list.

3.6.3 Member Function Documentation

3.6.3.1 void CellML_ConnectionList::append (const CellML_ConnectionList & *other*)

Append a list of connection's to this list.

Appends the contents of `other` to the end of this list, ensuring that the connection's added to the list have to same parent as this list.

Parameters:
other The list of connection's to add.

3.6.3.2 void CellML_ConnectionList::append (CellML_Connection & *newConnection*)

Append a connection to the list.

Adds `newConnection` to the end of the list. Appending the connections to the list will ensure that `newConnection` will have the same parent as this list.

Parameters:
newConnection The connection object to add to the end of this list.

3.6.3.3 bool CellML_ConnectionList::equals (const CellML_ConnectionList & *other*) const

Test two lists are equal.

Test the contents of `this` and `other` lists for equality. The list are deemed equal if they are both the same length and for each entry in `this` list there is an equal entry in the `other` list.

Parameters:
other The list to compare `this` list to.

Returns :
true if `this` list is equal to `other`; false otherwise.

3.6.3.4 const CellML_Connection& CellML_ConnectionList::get (const int *index*) const

Get a connection from the list.

Used to get a connection object from this list by its position in the list.

Parameters:
index The index of the desired connection (valid range is from 0 to **length**() (p. 39)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is outside the valid range for this connection list.

See also:
length() (p. 39)

3.6.3.5 const CellML_Model& CellML_ConnectionList::getParentModel () const

Gets the parent model of this list object.

Returns :
A constant reference to the parent CellML Model of this list object.

3.6.3.6 bool CellML_ConnectionList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.6.3.7 int CellML_ConnectionList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.6.3.8 CellML_ConnectionList& CellML_ConnectionList::operator= (const CellML_ConnectionList & *other*)

Assignment operator.

Parameters:
other The source to be assigned.

3.6.3.9 CellML_Connection CellML_ConnectionList::remove (const CellML_Connection & *oldConnection*)

Remove the given connection from the list.

Removes `oldConnection` from the list and returns it.

Parameters:
oldConnection The connection object to remove from the list.

Returns :
The connection removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldConnection` is not found in this list.

3.6.3.10 CellML_Connection CellML_ConnectionList::replace (CellML_Connection & *newConnection*, const CellML_Connection & *oldConnection*)

Replace a connection object with another.

Replaces `oldConnection` with `newConnection`. `newConnection` will be put into the same position in the list that is vacated by `oldConnection` - other than that this method is essentially the same as doing a remove(oldConnection) followed by a append(newConnection). This method will also ensure that `newConnection` has the same parent as this list.

Parameters:

newConnection The connection to add to the list.

oldConnection The connection to remove from the list.

Returns :

The connection removed from the list.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldConnection` is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

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

- CellML_ConnectionList.hpp

3.7 CellML_Equation Class Reference

A class for handling MathML equations.

Static Public Methods

- bool **checkMathMLElement** (const **CellML_MathMLElement** *math, const **CellML_Component** &component)

  Check a MathMLElement.

- **CellML_MathMLDocument** * **getAllMath** (const **CellML_Model** &model)

  Combine all model mathematics into a single MathML Document.

3.7.1 Detailed Description

A class for handling MathML equations.

This class simply provides static methods for convenient handling of MathML equations.

3.7.2 Member Function Documentation

3.7.2.1 bool CellML_Equation::checkMathMLElement (const CellML_MathMLElement * *math*, const CellML_Component & *component*) [static]

Check a MathMLElement.

Checks that all the `cn` elements in the given `math` element have valid `units` attributes in the CellML namespace, and that all the `ci` elements refer to valid `variable`s in the given `component`. **Note:** Does not check that `math` is valid MathML 2.0.

Parameters:

math The math to check.

component The **CellML_Component** (p. 4) to use to resolve the variables and units.

Returns :

`true` if the math element is valid; `false` otherwise.

3.7.2.2 CellML_MathMLDocument* CellML_Equation::getAllMath (const CellML_Model & *model*) [static]

Combine all model mathematics into a single MathML Document.

Create and return a MathML `document` that contains all the mathematics for the `model`, with all the variable references resolved into document-unique identifiers suitable for code generation.

Note: This should be extended to create a MathML Document which guarantees consistent units throughout. This could be done when resolving variables by adding equations which convert the "out" variable to the "in" variable in a dimensionally consistent manner. Of course, the equations would also need to be checked when mapping the result of an equation to an "in" variable somewhere else. **Too much work for Andre...**.

Parameters:

model The model to get the mathematics from.

Returns :

A MathML Document object that contains all the mathematics from the `model` with all identifiers resolved into document-wide unique identifiers suitable for code generation. The calling routine is responsible for the memory returned.

Exceptions:

CellML_Exception (p. 41) INVALID_CELLML_ERR: Raised if the model is not currently valid.

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

- CellML_Equation.hpp

3.8 CellML_Exception Class Reference

Encapsulate a general CellML error or warning.

Public Types

Enumerators for CellML Exceptions.

- enum
  ObjectType.

Public Methods

Constructors and assignment operator.

- **CellML_Exception** ()
  Default constructor.
- **CellML_Exception** (const int exCode, const char *message=0, const char *path=0)
  Constructor which takes an error code.
- **CellML_Exception** (const CellML_Exception &other)
  Copy constructor.
- CellML_Exception & **operator=** (const CellML_Exception &other)
  Assignment operator.

Destructor.

- **~CellML_Exception** ()
  Destructor for CellML_Exception.

- const char * **getMessage** () const
  Get the exception's message.
- const char * **getPath** () const
  Get the exception's path.

Public Attributes

Public variables.

- int **code**
  A code value, from the set defined by the class' enum, indicating the type of error that occured.

3.8.1 Detailed Description

Encapsulate a general CellML error or warning.

3.8.2 Member Enumeration Documentation

3.8.2.1 anonymous enum

ObjectType.

3.8.3 Constructor & Destructor Documentation

3.8.3.1 CellML_Exception::CellML_Exception ()

Default constructor.

3.8.3.2 CellML_Exception::CellML_Exception (const int *exCode*, const char * *message* = 0, const char * *path* = 0)

Constructor which takes an error code.

Parameters:

exCode The error code which indicates the exception.

message The message to associate with this exception.

path The path to associate with this exception.

See also:

getPath() (p. 43).

3.8.3.3 CellML_Exception::CellML_Exception (const CellML_Exception & *other*)

Copy constructor.

Parameters:

other The object to be copied.

3.8.3.4 CellML_Exception::~CellML_Exception ()

Destructor for CellML_Exception.

3.8.4 Member Function Documentation

3.8.4.1 const char* CellML_Exception::getMessage () const

Get the exception's message.

Returns a null-terminated character array.

Returns :

The message associated with this exception.

3.8.4.2 const char* CellML_Exception::getPath () const

Get the exception's path.

Returns a null-terminated character array. The path is meant to be used as an indication where the error occured in a model. For example: "Model - ModelName; Component - ComponentName; Units - UnitsName;"

Returns :

The path associated with this exception.

3.8.4.3 CellML_Exception& CellML_Exception::operator= (const CellML_Exception & *other*)

Assignment operator.

Parameters:

other The source to be assigned.

3.8.5 Member Data Documentation

3.8.5.1 int CellML_Exception::code

A code value, from the set defined by the class' enum, indicating the type of error that occured.

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

- CellML_Exception.hpp

3.9 CellML_Group Class Reference

The `CellML_Group` is used to group components into hierarchies and networks.

Public Methods

Constructors and assignment operator.

- **CellML_Group** (const **CellML_Model** &parentModel)
  Constructor for groups.
- **CellML_Group** (const CellML_Group &other)
  Copy constructor.
- CellML_Group & **operator=** (const CellML_Group &other)
  Assignment operator.

Destructor.

- ~**CellML_Group** ()
  Destructor for CellML_Group.

Equality.

- bool **equals** (const CellML_Group &other) const
  Check for equality of groups.

Get functions.

- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this group object.
- const **CellML_ComponentRefList** & **getComponentRefList** () const
  Get the component_ref list from this group.
- **CellML_ComponentRefList** & **getComponentRefList** ()
  Get the component_ref list from this group.
- const **CellML_RelationshipRefList** & **getRelationshipRefList** () const
  Get the relationship_ref list from this group.

- **CellML_RelationshipRefList** & **getRelationshipRefList** ()
  Get the relationship_ref list from this group.
- const **CellML_ComponentRef** & **getComponentRef** (const char *componentName) const
  Get a named component_ref from this group's hierarchy.

Cloning function.

- CellML_Group **clone** (const bool deep) const
  Returns a duplicate of this group.

Query functions.

- bool **isValid** () const
  Test whether this group is valid.
- bool **isHierarchyEncapsulation** () const
  Test if this group defines a CellML encapsulation hierarchy.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML Group into a DOM node.

Public Attributes

Factory methods.

- **CellML_ComponentRef createComponentRef** () const
  Create a component_ref.
- **CellML_RelationshipRef createRelationshipRef** () const
  Create a relationship_ref.

Private Methods

- **CellML_Group** ()
  Default constructor for CellML_Group.
- void **setParentModel** (const **CellML_Model** &model)
  Sets the parent model of this group.

Private Attributes

- const **CellML_Model** * **fParentModel**

  The parent of the group.

- GroupContent * **fContent**

  The content of the group.

Friends

- class **CellML_GroupList**

3.9.1 Detailed Description

The `CellML_Group` is used to group components into hierarchies and networks.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 CellML_Group::CellML_Group (const CellML_Model & *parentModel*)

Constructor for groups.

3.9.2.2 CellML_Group::CellML_Group (const CellML_Group & *other*)

Copy constructor.

The copy constructor will return a new `CellML_Group` object that has the same content object as the object being copied. If you want to have a new content object you need to use the `clone` method.

Parameters:
other The object to be copied.

See also:
clone() (p. 46).

3.9.2.3 CellML_Group::~CellML_Group ()

Destructor for CellML_Group.

3.9.2.4 CellML_Group::CellML_Group () `[private]`

Default constructor for CellML_Group.

3.9.3 Member Function Documentation

3.9.3.1 CellML_Group CellML_Group::clone (const bool *deep*) const

Returns a duplicate of this group.

This function serves as a generic copy constructor for groups.

Parameters:

deep If `true`, recursively clone the child objects of the group; if `false`, clone only the group itself.

Returns :

The duplicate group.

3.9.3.2 bool CellML_Group::equals (const CellML_Group & *other*) const

Check for equality of groups.

This will recursively descend through this group and the other group testing for equality.

Parameters:

other The group reference with which `this` object is compared.

Returns :

True if both groups are identical; return false otherwise.

3.9.3.3 void CellML_Group::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a group from the given DOM source node which should be a `group` element in the CellML namespace.

Parameters:

srcNode The DOM node containing a CellML group description.

Exceptions:

CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a group object from a DOM source.

DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.9.3.4 const CellML_ComponentRef& CellML_Group::getComponentRef (const char * *componentName*) const

Get a named component_ref from this group's hierarchy.

Convenience method to search this group's hierarchy for a component_ref element that has a component attribute with the given value.

Parameters:

componentName The value of the component attribute in the desired component_ref element.

Returns :

A constant reference to the component_ref element that has a component attribute with the value `componentName`, if one exists in this hierarchy.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if a component_ref element with the desired component attribute value does not exist in this hierarchy.

3.9.3.5 CellML_ComponentRefList& CellML_Group::getComponentRefList ()

Get the component_ref list from this group.

Returns :
A reference to this group's component_ref list.

3.9.3.6 const CellML_ComponentRefList& CellML_Group::getComponentRefList () const

Get the component_ref list from this group.

Returns :
A constant reference to this group's component_ref list.

3.9.3.7 const CellML_Model& CellML_Group::getParentModel () const

Gets the parent model of this group object.

Returns :
A constant reference to the parent CellML Model of this group object.

3.9.3.8 CellML_RelationshipRefList& CellML_Group::getRelationshipRefList ()

Get the relationship_ref list from this group.

Returns :
A reference to this group's relationship_ref list.

3.9.3.9 const CellML_RelationshipRefList& CellML_Group::getRelationshipRefList () const

Get the relationship_ref list from this group.

Returns :
A constant reference to this group's relationship_ref list.

3.9.3.10 bool CellML_Group::isHierarchyEncapsulation () const

Test if this group defines a CellML encapsulation hierarchy.

Convenience method to determine if this group contains a relationship_ref element with a relationship attribute with the value of "encapsulation" in the CellML namespace.

Returns :
True if this group defines a CellML encapsulation hierarchy; false otherwise.

3.9.3.11 bool CellML_Group::isValid () const

Test whether this group is valid.

What does this mean for a CellML Group ?? This should only be called once the group has been fully populated and the calling routine wants to check the validity of the current contents of the group.

Returns :
True if the group is valid; false otherwise.

3.9.3.12 CellML_Group& CellML_Group::operator= (const CellML_Group & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the `clone` method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 46).

3.9.3.13 void CellML_Group::setParentModel (const CellML_Model & *model*) [private]

Sets the parent model of this group.

Parameters:
model The parent model of this group.

3.9.3.14 DOMNode* CellML_Group::toNode () const

Translate this CellML Group into a DOM node.

Returns :
The DOM Node created which represents the current contents of this group object.

3.9.4 Member Data Documentation

3.9.4.1 CellML_ComponentRef CellML_Group::createComponentRef() const

Create a component_ref.

Creates an empty **CellML_ComponentRef** (p. 18) object as a child of this group.

Returns :
The component_ref object created.

3.9.4.2 CellML_RelationshipRef CellML_Group::createRelationshipRef() const

Create a relationship_ref.

Creates an empty **CellML_RelationshipRef** (p. 125) object as a child of this group.

Returns :
The relationship_ref object created.

3.9.4.3 GroupContent* CellML_Group::fContent [private]

The content of the group.

3.9.4.4 const CellML_Model* CellML_Group::fParentModel [private]

The parent of the group.

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

- CellML_Group.hpp

3.10 CellML_GroupList Class Reference

A class for representing lists of **CellML_Group** (p. 44)s.

Public Methods

Constructors and assignment operators.

- **CellML_GroupList** (const **CellML_Model** &parentModel)
  Constructor for lists belonging to models.
- **CellML_GroupList** (const CellML_GroupList &other)
  Copy constructor.
- CellML_GroupList & **operator=** (const CellML_GroupList &other)
  Assignment operator.

Destructor.

- **~CellML_GroupList** ()
  Destructor for CellML_GroupList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.

- bool **equals** (const CellML_GroupList &other) const
  Test two lists are equal.
- const **CellML_Group** & **get** (const int index) const
  Get a group from the list.
- void **append** (**CellML_Group** &newGroup)
  Append a group to the list.
- void **append** (const CellML_GroupList &other)
  Append a list of group's to this list.
- **CellML_Group remove** (const **CellML_Group** &oldGroup)
  Remove the given group from the list.
- **CellML_Group replace** (**CellML_Group** &newGroup, const **CellML_Group** &oldGroup)
  Replace a group object with another.
- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this list object.

Private Methods

- **CellML_GroupList** ()
  Default constructor.

3.10.1 Detailed Description

A class for representing lists of **CellML_Group** (p. 44)s.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 CellML_GroupList::CellML_GroupList (const CellML_Model & *parentModel*)

Constructor for lists belonging to models.

Parameters:
parentModel The parent of this list.

3.10.2.2 CellML_GroupList::CellML_GroupList (const CellML_GroupList & *other*)

Copy constructor.

Copies only references to objects in the list, not the actual objects. Note that this does not change the parent of the list or the elements of the list. If you wish to create a copy of a list to initialise a new object you should first create an empty list and then append the existing list to the new list.

Parameters:
other The list to be copied.

See also:
append(const CellML_GroupList&) (p. 52)

3.10.2.3 CellML_GroupList::~CellML_GroupList ()

Destructor for CellML_GroupList.

3.10.2.4 CellML_GroupList::CellML_GroupList () [private]

Default constructor.

Will construct an empty list.

3.10.3 Member Function Documentation

3.10.3.1 void CellML_GroupList::append (const CellML_GroupList & *other*)

Append a list of group's to this list.

Appends the contents of `other` to the end of this list, ensuring that the group's added to the list have to same parent as this list.

Parameters:
other The list of group's to add.

3.10.3.2 void CellML_GroupList::append (CellML_Group & *newGroup*)

Append a group to the list.

Adds `newGroup` to the end of the list. Appending the groups to the list will ensure that `newGroup` will have the same parent as this list.

Parameters:
newGroup The group object to add to the end of this list.

3.10.3.3 bool CellML_GroupList::equals (const CellML_GroupList & *other*) const

Test two lists are equal.

Test the contents of `this` and `other` lists for equality. The list are deemed equal if they are both the same length and for each entry in `this` list there is an equal entry in the `other` list.

Parameters:
other The list to compare `this` list to.

Returns :
true if `this` list is equal to `other`; false otherwise.

3.10.3.4 const CellML_Group& CellML_GroupList::get (const int *index*) const

Get a group from the list.

Used to get a group object from this list by its position in the list.

Parameters:
index The index of the desired group (valid range is from 0 to **length**() (p. 53)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is outside the valid range for this group list.

See also:
length() (p. 53)

3.10.3.5 const CellML_Model& CellML_GroupList::getParentModel () const

Gets the parent model of this list object.

Returns :
A constant reference to the parent CellML Model of this list object.

3.10.3.6 bool CellML_GroupList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.10.3.7 int CellML_GroupList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.10.3.8 CellML_GroupList& CellML_GroupList::operator= (const CellML_GroupList & *other*)

Assignment operator.

Parameters:
other The source to be assigned.

3.10.3.9 CellML_Group CellML_GroupList::remove (const CellML_Group & *oldGroup*)

Remove the given group from the list.

Removes `oldGroup` from the list and returns it.

Parameters:
oldGroup The group object to remove from the list.

Returns :
The group removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldGroup` is not found in this list.

3.10.3.10 CellML_Group CellML_GroupList::replace (CellML_Group & *newGroup*, const CellML_Group & *oldGroup*)

Replace a group object with another.

Replaces `oldGroup` with `newGroup`. `newGroup` will be put into the same position in the list that is vacated by `oldGroup` - other than that this method is essentially the same as doing a remove(oldGroup) followed by a append(newGroup). This method will also ensure that `newGroup` has the same parent as this list.

Parameters:
newGroup The group to add to the list.

oldGroup The group to remove from the list.

Returns :
The group removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldGroup` is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

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

- CellML_GroupList.hpp

3.11 CellML_MapComponents Class Reference

`CellML_MapComponents` are used to specify the components at either ends of a **CellML_Connection** (p. 29).

Public Methods

Constructors and assignment operators.

- **CellML_MapComponents** (const **CellML_Connection** &parentConnection)

Connection constructor.

- **CellML_MapComponents** (const CellML_MapComponents &other)
  Copy constructor.
- CellML_MapComponents & **operator=** (const CellML_MapComponents &other)
  Assignment operator.

Destructor.

- **~CellML_MapComponents** ()
  Destructor for CellML_MapComponents.

Equality.

- bool **equals** (const CellML_MapComponents &other) const
  Check for equality of map_components.

Get functions.

- const **CellML_Connection** & **getParentConnection** () const
  Gets the parent connection of this map_components object.
- const char * **getComponent1Name** () const
  Gets the name of the first component referenced.
- const **CellML_Component** & **getComponent1** () const
  Gets a reference to the first component referenced.
- const char * **getComponent2Name** () const
  Gets the name of the second component referenced.
- const **CellML_Component** & **getComponent2** () const
  Gets a reference to the second component referenced.

Cloning function.

- CellML_MapComponents **clone** (const bool deep) const
  Returns a duplicate of this map_components object.

Query functions.

- bool **isValid** () const
  Test whether this map_components is valid.

Set functions.

- void **setComponent1Name** (const char *name)

Sets the name of the first component referenced.

- void **setComponent2Name** (const char *name)
  Sets the name of the second component referenced.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML MapComponents into a DOM node.

Private Methods

- **CellML_MapComponents** ()
  Default constructor for CellML_MapComponents.
- void **setParentConnection** (const **CellML_Connection** &connection)
  Sets the parent connection of this map_components object.

Private Attributes

- const **CellML_Connection** * **fParentConnection**
  The parent connection.
- MapComponentsContent * **fContent**
  The content of the map components.

Friends

- class **CellML_Connection**

3.11.1 Detailed Description

`CellML_MapComponents` are used to specify the components at either ends of a **CellML_Connection** (p. 29).

3.11.2 Constructor & Destructor Documentation

3.11.2.1 CellML_MapComponents::CellML_MapComponents (const CellML_Connection & *parentConnection*)

Connection constructor.

Parameters:
parentConnection The parent CellML connection of this map_components.

3.11.2.2 CellML_MapComponents::CellML_MapComponents (const CellML_MapComponents & *other*)

Copy constructor.

The copy constructor will return a new `CellML_MapComponents` object that has the same content object as the object being copied. If you want to have a new content object you need to use the `clone` method.

Parameters:
other The object to be copied.

See also:
clone() (p. 57).

3.11.2.3 CellML_MapComponents::~CellML_MapComponents ()

Destructor for CellML_MapComponents.

3.11.2.4 CellML_MapComponents::CellML_MapComponents () [private]

Default constructor for CellML_MapComponents.

3.11.3 Member Function Documentation

3.11.3.1 CellML_MapComponents CellML_MapComponents::clone (const bool *deep*) const

Returns a duplicate of this map_components object.

This function serves as a generic copy constructor for map_components. Cloning a map_components results in a new content object being created and initialised with the contents of the map_components being cloned.

Parameters:
deep If `true`, recursively clone the child objects of the map_components; if `false`, clone only the map_components itself.

Returns :
The duplicate map_components.

3.11.3.2 bool CellML_MapComponents::equals (const CellML_MapComponents & *other*) const

Check for equality of map_components.

This will test for equality between `this` map_components and `other`.

Parameters:
other The map_components reference with which `this` object is compared.

Returns :
True if both map_components are identical; return false otherwise.

3.11.3.3 void CellML_MapComponents::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a map_components from the given DOM source node which should be a `map_components` element in the CellML namespace.

Parameters:

srcNode The DOM node containing a CellML map_components description.

Exceptions:

CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a map_components object from a DOM source.

DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.11.3.4 const CellML_Component& CellML_MapComponents::getComponent1 () const

Gets a reference to the first component referenced.

Convenience method to search the tree for the appropriate component object.

Returns :

A constant reference to the first component referenced.

Exceptions:

CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the this map_components' component_1 name is an illegal CellML identifier.

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if the component referenced by this map_components' can not be found in the current model.

3.11.3.5 const char* CellML_MapComponents::getComponent1Name () const

Gets the name of the first component referenced.

Returns :

A null-terminated string representation of the name of the component_1 attribute of this map_components.

3.11.3.6 const CellML_Component& CellML_MapComponents::getComponent2 () const

Gets a reference to the second component referenced.

Convenience method to search the tree for the appropriate component object.

Returns :

A constant reference to the second component referenced.

Exceptions:

CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the this map_components' component_1 name is an illegal CellML identifier.

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if the component referenced by this map_components' can not be found in the current model.

3.11.3.7 const char* CellML_MapComponents::getComponent2Name () const

Gets the name of the second component referenced.

Returns :
A null-terminated string representation of the name of the component_2 attribute of this map_components.

3.11.3.8 const CellML_Connection& CellML_MapComponents::getParentConnection () const

Gets the parent connection of this map_components object.

Returns :
A constant reference to the parent CellML Connection of this map_components object.

3.11.3.9 bool CellML_MapComponents::isValid () const

Test whether this map_components is valid.

What does this mean for a CellML MapComponents ?? This should only be called once the map_components has been fully populated and the calling routine wants to check the validity of the current contents of the map_components.

Returns :
True if the map_components is valid; false otherwise.

3.11.3.10 CellML_MapComponents& CellML_MapComponents::operator= (const CellML_MapComponents & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the `clone` method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 57).

3.11.3.11 void CellML_MapComponents::setComponent1Name (const char * *name*)

Sets the name of the first component referenced.

Takes a copy of the `name` so it can be safely freed by the calling routine.

Parameters:
name The name of the first component to be referenced.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the `name` is an illegal CellML identifier.

3.11.3.12 void CellML_MapComponents::setComponent2Name (const char * *name*)

Sets the name of the second component referenced.

Takes a copy of the `name` so it can be safely freed by the calling routine.

Parameters:
name The name of the second component to be referenced.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the `name` is an illegal CellML identifier.

3.11.3.13 void CellML_MapComponents::setParentConnection (const CellML_Connection & *connection*) [private]

Sets the parent connection of this map_components object.

3.11.3.14 DOMNode* CellML_MapComponents::toNode () const

Translate this CellML MapComponents into a DOM node.

Returns :
The DOM Node created which represents the current contents of this map_components object.

3.11.4 Member Data Documentation

3.11.4.1 MapComponentsContent* CellML_MapComponents::fContent [private]

The content of the map components.

3.11.4.2 const CellML_Connection* CellML_MapComponents::fParentConnection [private]

The parent connection.

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

- CellML_MapComponents.hpp

3.12 CellML_MapVariables Class Reference

`CellML_MapVariables` are used to specify the variables at either ends of a **CellML_Connection** (p. 29).

Public Methods

Constructors and assignment operators.

- **CellML_MapVariables** (const **CellML_Connection** &parentConnection)
  Connection constructor.

- **CellML_MapVariables** (const CellML_MapVariables &other)
  Copy constructor.
- CellML_MapVariables & **operator=** (const CellML_MapVariables &other)
  Assignment operator.

Destructor.

- ~**CellML_MapVariables** ()
  Destructor for CellML_MapVariables.

Equality.

- bool **equals** (const CellML_MapVariables &other) const
  Check for equality of map_variables'.

Get functions.

- const **CellML_Connection** & **getParentConnection** () const
  Gets the parent connection of this map_variables object.
- const char * **getVariable1Name** () const
  Gets the name of the first variable referenced.
- const **CellML_Variable** & **getVariable1** () const
  Gets a reference to the first variable referenced.
- const char * **getVariable2Name** () const
  Gets the name of the second variable referenced.
- const **CellML_Variable** & **getVariable2** () const
  Gets a reference to the second variable referenced.

Cloning function.

- CellML_MapVariables **clone** (const bool deep) const
  Returns a duplicate of this map_variables object.

Query functions.

- bool **isValid** () const
  Test whether this map_variables is valid.

Set functions.

- void **setVariable1Name** (const char *name)
  Sets the name of the first variable referenced.

- void **setVariable2Name** (const char *name)
  Sets the name of the second variable referenced.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML MapVariables into a DOM node.

Private Methods

- **CellML_MapVariables** ()
  Default constructor for CellML_MapVariables.
- void **setParentConnection** (const **CellML_Connection** &connection)
  Sets the parent connection of this map_variables object.

Private Attributes

- const **CellML_Connection** * **fParentConnection**
  The parent connection.
- MapVariablesContent * **fContent**
  The content of the map_variables.

Friends

- class **CellML_MapVariablesList**

3.12.1 Detailed Description

`CellML_MapVariables` are used to specify the variables at either ends of a **CellML_Connection** (p. 29).

3.12.2 Constructor & Destructor Documentation

3.12.2.1 CellML_MapVariables::CellML_MapVariables (const CellML_Connection & *parentConnection*)

Connection constructor.

Parameters:
parentConnection The parent CellML connection of this map_variables.

3.12.2.2 CellML_MapVariables::CellML_MapVariables (const CellML_MapVariables & *other*)

Copy constructor.

The copy constructor will return a new `CellML_MapVariables` object that has the same content object as the object being copied. If you want to have a new content object you need to use the `clone` method.

Parameters:
other The object to be copied.

See also:
clone() (p. 63).

3.12.2.3 CellML_MapVariables::~CellML_MapVariables ()

Destructor for CellML_MapVariables.

3.12.2.4 CellML_MapVariables::CellML_MapVariables () [private]

Default constructor for CellML_MapVariables.

3.12.3 Member Function Documentation

3.12.3.1 CellML_MapVariables CellML_MapVariables::clone (const bool *deep*) const

Returns a duplicate of this map_variables object.

This function serves as a generic copy constructor for map_variables. Cloning a map_variables results in a new content object being created and initialised with the contents of the map_variables being cloned.

Parameters:
deep If `true`, recursively clone the child objects of the map_variables; if `false`, clone only the map_variables itself.

Returns :
The duplicate map_variables.

3.12.3.2 bool CellML_MapVariables::equals (const CellML_MapVariables & *other*) const

Check for equality of map_variables'.

This will test for equality between `this` map_variables and `other`.

Parameters:
other The map_variables reference with which `this` object is compared.

Returns :
True if both map_variables are identical; return false otherwise.

3.12.3.3 void CellML_MapVariables::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a map_variables from the given DOM source node which should be a `map_variables` element in the CellML namespace.

Parameters:

srcNode The DOM node containing a CellML map_variables description.

Exceptions:

CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a map_variables object from a DOM source.

DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.12.3.4 const CellML_Connection& CellML_MapVariables::getParentConnection () const

Gets the parent connection of this map_variables object.

Returns :

A constant reference to the parent CellML Connection of this map_variables object.

3.12.3.5 const CellML_Variable& CellML_MapVariables::getVariable1 () const

Gets a reference to the first variable referenced.

Convenience method to search the tree for the appropriate variable object.

Returns :

A constant reference to the first variable referenced.

Exceptions:

CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the this map_variables' variable_1 name is an illegal CellML identifier.

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if the variable referenced by this map_variables' variable_1 attribute can not be found in the component referenced by the parent connection's map_components' component_1; or if the component_1 itself can not be found; or if the the map_components can not be found.

3.12.3.6 const char* CellML_MapVariables::getVariable1Name () const

Gets the name of the first variable referenced.

Returns :

A null-terminated string representation of the name of the variable_1 attribute of this map_variables.

3.12.3.7 const CellML_Variable& CellML_MapVariables::getVariable2 () const

Gets a reference to the second variable referenced.

Convenience method to search the tree for the appropriate variable object.

Returns :
A constant reference to the second variable referenced.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the this map_variables' variable_2 name is an illegal CellML identifier.
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if the variable referenced by this map_variables' variable_1 attribute can not be found in the component referenced by the parent connection's map_components' component_1; or if the component_1 itself can not be found; or if the the map_components can not be found.

3.12.3.8 const char* CellML_MapVariables::getVariable2Name () const

Gets the name of the second variable referenced.

Returns :
A null-terminated string representation of the name of the variable_1 attribute of this map_variables.

3.12.3.9 bool CellML_MapVariables::isValid () const

Test whether this map_variables is valid.

What does this mean for a CellML MapVariables ?? This should only be called once the map_variables has been fully populated and the calling routine wants to check the validity of the current contents of the map_variables.

Returns :
True if the map_variables is valid; false otherwise.

3.12.3.10 CellML_MapVariables& CellML_MapVariables::operator= (const CellML_MapVariables & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the `clone` method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 63).

3.12.3.11 void CellML_MapVariables::setParentConnection (const CellML_Connection & *connection*) [private]

Sets the parent connection of this map_variables object.

3.12.3.12 void CellML_MapVariables::setVariable1Name (const char * *name*)

Sets the name of the first variable referenced.

Takes a copy of the `name` so it can be safely freed by the calling routine.

Parameters:
name The name of the first variable to be referenced.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the `name` is an illegal CellML identifier.

3.12.3.13 void CellML_MapVariables::setVariable2Name (const char * *name*)

Sets the name of the second variable referenced.

Takes a copy of the `name` so it can be safely freed by the calling routine.

Parameters:
name The name of the second variable to be referenced.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the `name` is an illegal CellML identifier.

3.12.3.14 DOMNode* CellML_MapVariables::toNode () const

Translate this CellML MapVariables into a DOM node.

Returns :
The DOM Node created which represents the current contents of this map_variables object.

3.12.4 Member Data Documentation

3.12.4.1 MapVariablesContent* CellML_MapVariables::fContent [private]

The content of the map_variables.

3.12.4.2 const CellML_Connection* CellML_MapVariables::fParentConnection [private]

The parent connection.

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

- CellML_MapVariables.hpp

3.13 CellML_MapVariablesList Class Reference

A class for representing lists of **CellML_MapVariables** (p. 60)s.

Public Methods

Constructors and assignment operators.

- **CellML_MapVariablesList** (const **CellML_Connection** &parentConnection)
  Constructor for lists belonging to connections.
- **CellML_MapVariablesList** (const CellML_MapVariablesList &other)
  Copy constructor.
- CellML_MapVariablesList & **operator=** (const CellML_MapVariablesList &other)
  Assignment operator.

Destructor.

- **~CellML_MapVariablesList** ()
  Destructor for CellML_MapVariablesList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_MapVariablesList &other) const
  Test two lists are equal.
- const **CellML_MapVariables** & **get** (const int index) const
  Get a map_variables from the list.
- void **append** (**CellML_MapVariables** &newMapVariables)
  Append a map_variables to the list.
- void **append** (const CellML_MapVariablesList &other)
  Append a list of map_variables's to this list.
- **CellML_MapVariables remove** (const **CellML_MapVariables** &oldMapVariables)
  Remove the given map_variables from the list.
- **CellML_MapVariables replace** (**CellML_MapVariables** &newMapVariables, const **CellML_MapVariables** &oldMapVariables)
  Replace a map_variables object with another.
- const **CellML_Connection** & **getParentConnection** () const
  Gets the parent connection of this list object.

Private Methods

- **CellML_MapVariablesList** ()
  Default constructor.

3.13.1 Detailed Description

A class for representing lists of **CellML_MapVariables** (p. 60)s.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 CellML_MapVariablesList::CellML_MapVariablesList (const CellML_Connection & *parentConnection*)

Constructor for lists belonging to connections.

Parameters:
parentConnection The parent of this list.

3.13.2.2 CellML_MapVariablesList::CellML_MapVariablesList (const CellML_MapVariablesList & *other*)

Copy constructor.

Copies only references to objects in the list, not the actual objects. Note that this does not change the parent of the list or the elements of the list. If you wish to create a copy of a list to initialise a new object you should first create an empty list and then append the existing list to the new list.

Parameters:
other The list to be copied.

See also:
append(const CellML_MapVariablesList&) (p. 68)

3.13.2.3 CellML_MapVariablesList::~CellML_MapVariablesList ()

Destructor for CellML_MapVariablesList.

3.13.2.4 CellML_MapVariablesList::CellML_MapVariablesList () [private]

Default constructor.

Will construct an empty list.

3.13.3 Member Function Documentation

3.13.3.1 void CellML_MapVariablesList::append (const CellML_MapVariablesList & *other*)

Append a list of map_variables's to this list.

Appends the contents of `other` to the end of this list, ensuring that the map_variables' added to the list have to same parent as this list.

Parameters:
other The list of map_variables' to add.

3.13.3.2 void CellML_MapVariablesList::append (CellML_MapVariables & *newMapVariables*)

Append a map_variables to the list.

Adds `newMapVariables` to the end of the list. Appending the map_variabless to the list will ensure that `newMapVariables` will have the same parent as this list.

Parameters:
newMapVariables The map_variables object to add to the end of this list.

3.13.3.3 bool CellML_MapVariablesList::equals (const CellML_MapVariablesList & *other*) const

Test two lists are equal.

Test the contents of `this` and `other` lists for equality. The list are deemed equal if they are both the same length and for each entry in `this` list there is an equal entry in the `other` list.

Parameters:
other The list to compare `this` list to.

Returns :
true if `this` list is equal to `other`; false otherwise.

3.13.3.4 const CellML_MapVariables& CellML_MapVariablesList::get (const int *index*) const

Get a map_variables from the list.

Used to get a map_variables object from this list by its position in the list.

Parameters:
index The index of the desired map_variables (valid range is from 0 to **length**() (p. 70)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is outside the valid range for this map_variables list.

See also:
length() (p. 70)

3.13.3.5 const CellML_Connection& CellML_MapVariablesList::getParentConnection () const

Gets the parent connection of this list object.

Returns :
A constant reference to the parent CellML Connection of this list object.

3.13.3.6 bool CellML_MapVariablesList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.13.3.7 int CellML_MapVariablesList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.13.3.8 CellML_MapVariablesList& CellML_MapVariablesList::operator= (const CellML_MapVariablesList & *other*)

Assignment operator.

Parameters:
other The source to be assigned.

3.13.3.9 CellML_MapVariables CellML_MapVariablesList::remove (const CellML_MapVariables & *oldMapVariables*)

Remove the given map_variables from the list.

Removes `oldMapVariables` from the list and returns it.

Parameters:
oldMapVariables The map_variables object to remove from the list.

Returns :
The map_variables removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldMapVariables` is not found in this list.

3.13.3.10 CellML_MapVariables CellML_MapVariablesList::replace (CellML_MapVariables & *newMapVariables*, const CellML_MapVariables & *oldMapVariables*)

Replace a map_variables object with another.

Replaces `oldMapVariables` with `newMapVariables`. `newMapVariables` will be put into the same position in the list that is vacated by `oldMapVariables` - other than that this method is essentially the same as doing a remove(oldMapVariables) followed by a append(newMapVariables). This method will also ensure that `newMapVariables` has the same parent as this list.

Parameters:
newMapVariables The map_variables to add to the list.

oldMapVariables The map_variables to remove from the list.

Returns :
The map_variables removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `oldMapVariables` is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

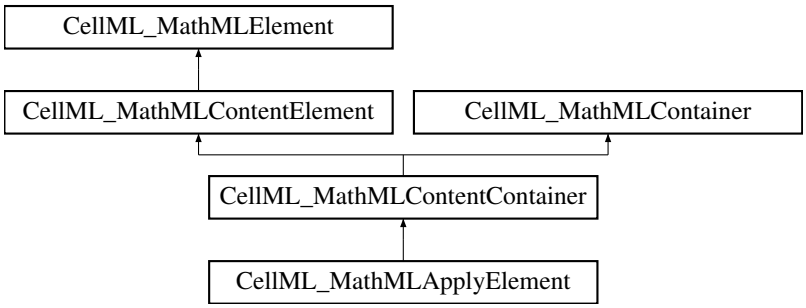
The documentation for this class was generated from the following file:

- CellML_MapVariablesList.hpp

3.14 CellML_MathMLApplyElement Class Reference

The `apply` element allows a function or operator to be applied to its arguments.

Inheritance diagram for CellML_MathMLApplyElement::



Public Methods

- **CellML_MathMLApplyElement** (const **CellML_MathMLMathElement** *ownerMathElement)
  Constructor.
- ~**CellML_MathMLApplyElement** ()
  Destructor.
- **CellML_MathMLElement** * **getOperator** () const
  Get the operator element.
- void **setOperator** (**CellML_MathMLElement** *op)
  Set the operator element.
- **CellML_MathMLElement** * **getLogBase** () const
  Get the log base element.
- void **setLogBase** (**CellML_MathMLElement** *base)
  Set the log base element.

- **CellML_MathMLElement** * **getLowLimit** ()
  Get the lowlimit element.
- void **setLowLimit** (**CellML_MathMLElement** *lowLimit)
  Set the lowlimit element.
- **CellML_MathMLElement** * **getUpLimit** ()
  Get the uplimit element.
- void **setUpLimit** (**CellML_MathMLElement** *upLimit)
  Set the uplimit element.
- void **fromNode** (const DOMNode *node)
  Translate a DOM Node into a `apply` *element.*

Private Methods

- **CellML_MathMLApplyElement** ()
  Constructor.
- **CellML_MathMLApplyElement** (const CellML_MathMLApplyElement *other)
  Copy constructor.

3.14.1 Detailed Description

The `apply` element allows a function or operator to be applied to its arguments.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 CellML_MathMLApplyElement::CellML_MathMLApplyElement (const CellML_MathMLMathElement * *ownerMathElement*)

Constructor.

3.14.2.2 CellML_MathMLApplyElement::~CellML_MathMLApplyElement ()

Destructor.

3.14.2.3 CellML_MathMLApplyElement::CellML_MathMLApplyElement () `[private]`

Constructor.

3.14.2.4 CellML_MathMLApplyElement::CellML_MathMLApplyElement (const CellML_MathMLApplyElement * *other*) `[private]`

Copy constructor.

3.14.3 Member Function Documentation

3.14.3.1 void CellML_MathMLApplyElement::fromNode (const DOMNode * *node*)

Translate a DOM Node into a `apply` element.

Parameters:
node The node to translate.

3.14.3.2 CellML_MathMLElement* CellML_MathMLApplyElement::getLogBase () const

Get the log base element.

Returns :
The MathML element representing the `logbase` when the `log` operator is being applied; or null if there is no `logbase`.

3.14.3.3 CellML_MathMLElement* CellML_MathMLApplyElement::getLowLimit ()

Get the lowlimit element.

Returns :
The `lowlimit` child element of this node (if any). This expresses, for instance, the lower limit of integration if this is an `apply` element whose first child is a `int`.

3.14.3.4 CellML_MathMLElement* CellML_MathMLApplyElement::getOperator () const

Get the operator element.

Returns :
The MathML element representing the function or operator that is applied to the list of arguments.

3.14.3.5 CellML_MathMLElement* CellML_MathMLApplyElement::getUpLimit ()

Get the uplimit element.

Returns :
The `uplimit` child element of this node (if any). This expresses, for instance, the upper limit of integration if this is an `apply` element whose first child is a `int`.

3.14.3.6 void CellML_MathMLApplyElement::setLogBase (CellML_MathMLElement * *base*)

Set the log base element.

Parameters:
base The MathML element representing the `logbase` when the `log` operator is being applied.

3.14.3.7 void CellML_MathMLApplyElement::setLowLimit (CellML_MathMLElement * *lowLimit*)

Set the lowlimit element.

Parameters:

lowLimit The `lowlimit` child element of this node. This expresses, for instance, the lower limit of integration if this is an `apply` element whose first child is a `int`.

Exceptions:

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child `lowlimit` element. In particular, raised if this element is not an `apply` element whose first child is an `int`, `sum`, `product`, or `limit` element.

3.14.3.8 void CellML_MathMLApplyElement::setOperator (CellML_MathMLElement * *op*)

Set the operator element.

Parameters:

op The MathML element representing the function or operator that is applied to the list of arguments.

3.14.3.9 void CellML_MathMLApplyElement::setUpLimit (CellML_MathMLElement * *upLimit*)

Set the uplimit element.

Parameters:

upLimit The `uplimit` child element of this node. This expresses, for instance, the upper limit of integration if this is an `apply` element whose first child is a `int`.

Exceptions:

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child `uplimit` element. In particular, raised if this element is not an `apply` element whose first child is an `int`, `sum`, `product`, or `limit` element.

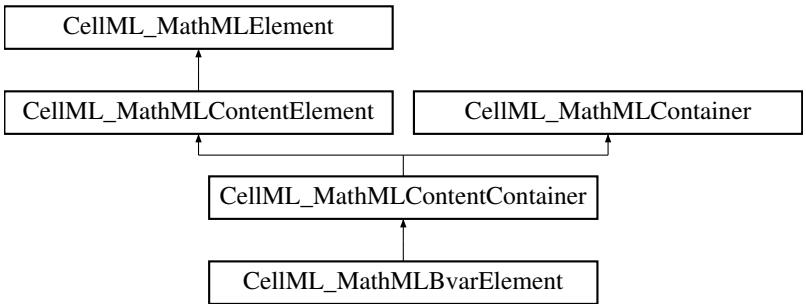
The documentation for this class was generated from the following file:

- CellML_MathMLApplyElement.hpp

3.15 CellML_MathMLBvarElement Class Reference

This interface represents the MathML bound variable element `bvar`.

Inheritance diagram for CellML_MathMLBvarElement::



Public Methods

- **CellML_MathMLBvarElement** (const **CellML_MathMLMathElement** *ownerMathElement)
  Constructor.
- **~CellML_MathMLBvarElement** ()
  Destructor.

Private Methods

- **CellML_MathMLBvarElement** ()
  Constructor.
- **CellML_MathMLBvarElement** (const CellML_MathMLBvarElement *other)
  Copy constructor.

3.15.1 Detailed Description

This interface represents the MathML bound variable element `bvar`.

The interface currently provides no functionality beyond that of `MathMLContentContainer`, but is useful for defining the type of bound variable access functions.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 CellML_MathMLBvarElement::CellML_MathMLBvarElement (const CellML_MathMLMathElement * *ownerMathElement*)

Constructor.

3.15.2.2 CellML_MathMLBvarElement::~CellML_MathMLBvarElement ()

Destructor.

3.15.2.3 CellML_MathMLBvarElement::CellML_MathMLBvarElement () `[private]`

Constructor.

3.15.2.4 CellML_MathMLBvarElement::CellML_MathMLBvarElement (const CellML_MathMLBvarElement * *other*) [private]

Copy constructor.

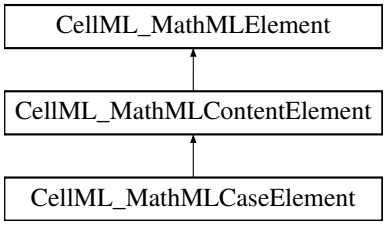
The documentation for this class was generated from the following file:

- CellML_MathMLBvarElement.hpp

3.16 CellML_MathMLCaseElement Class Reference

The `piece` element represents one of a sequence of cases used in the piecewise definition of a function.

Inheritance diagram for CellML_MathMLCaseElement::



Public Methods

- **CellML_MathMLCaseElement** (const **CellML_MathMLMathElement** *ownerMathElement)

  Constructor.

- ~**CellML_MathMLCaseElement** ()

  Destructor.

- **CellML_MathMLContentElement** * **getCaseCondition** () const

  Get the case condition.

- void **setCaseCondition** (const **CellML_MathMLContentElement** *caseCondition)

  Set the case condition.

- **CellML_MathMLContentElement** * **getCaseValue** () const

  Get the case value.

- void **setCaseValue** (const **CellML_MathMLContentElement** *caseValue)

  Set the case value.

Private Methods

- **CellML_MathMLCaseElement** ()

  Constructor.

- **CellML_MathMLCaseElement** (const CellML_MathMLCaseElement *other)
  Copy constructor.

3.16.1 Detailed Description

The `piece` element represents one of a sequence of cases used in the piecewise definition of a function.

It contains two child elements, each represented by a `MathMLContentElement`. The first child determines the subset of the domain affected, normally by giving a condition to be satisfied. The second gives the value of the function over the indicated subset of its domain.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 CellML_MathMLCaseElement::CellML_MathMLCaseElement (const CellML_MathMLMathElement * *ownerMathElement*)

Constructor.

3.16.2.2 CellML_MathMLCaseElement::~CellML_MathMLCaseElement ()

Destructor.

3.16.2.3 CellML_MathMLCaseElement::CellML_MathMLCaseElement () `[private]`

Constructor.

3.16.2.4 CellML_MathMLCaseElement::CellML_MathMLCaseElement (const CellML_MathMLCaseElement * *other*) `[private]`

Copy constructor.

3.16.3 Member Function Documentation

3.16.3.1 CellML_MathMLContentElement* CellML_MathMLCaseElement::getCaseCondition () const

Get the case condition.

Returns :
The `MathMLContentElement` representing the condition to be satisfied in order for this branch of the piecewise definition to be used.

3.16.3.2 CellML_MathMLContentElement* CellML_MathMLCaseElement::getCaseValue () const

Get the case value.

Returns :
The `MathMLContentElement` representing the value to be taken by the piecewise function when the condition described by the case condition is true.

3.16.3.3 void CellML_MathMLCaseElement::setCaseCondition (const CellML_MathMLContentElement * *caseCondition*)

Set the case condition.

Parameters:

caseCondition The `MathMLContentElement` representing the condition to be satisfied in order for this branch of the piecewise definition to be used.

3.16.3.4 void CellML_MathMLCaseElement::setCaseValue (const CellML_MathMLContentElement * *caseValue*)

Set the case value.

Parameters:

caseValue The `MathMLContentElement` representing the value to be taken by the piecewise function when the condition described by the case condition is true.

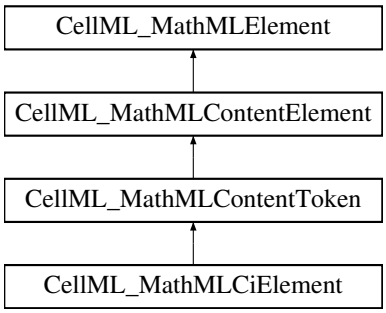
The documentation for this class was generated from the following file:

- CellML_MathMLCaseElement.hpp

3.17 CellML_MathMLCiElement Class Reference

The `ci` element is used to specify a symbolic name.

Inheritance diagram for CellML_MathMLCiElement::



Public Methods

- **CellML_MathMLCiElement** (const **CellML_MathMLMathElement** *ownerMathElement, const char *identifier)

  Constructor.

- **~CellML_MathMLCiElement** ()

  Destructor.

- const char * **getType** () const

  Get the type.

- void **setType** (const char *type)
  Set the type.
- const char * **getIdentifier** () const
  Get the identifier name.
- void **setIdentifier** (const char *identifier)
  Set the identifier name.
- bool **isInitialValueSet** () const
  Check if this `ci` *element has had its initial value set.*
- float **getInitialValue** () const
  Get the initial value for this `ci` *element.*
- void **setInitialValue** (float value)
  Set the initial value for this `ci` *element.*

Private Methods

- **CellML_MathMLCiElement** ()
  Constructor.
- **CellML_MathMLCiElement** (const CellML_MathMLCiElement *other)
  Copy constructor.

3.17.1 Detailed Description

The `ci` element is used to specify a symbolic name.

3.17.2 Constructor & Destructor Documentation

3.17.2.1 CellML_MathMLCiElement::CellML_MathMLCiElement (const CellML_MathMLMathElement * *ownerMathElement*, const char * *identifier*)

Constructor.

3.17.2.2 CellML_MathMLCiElement::~CellML_MathMLCiElement ()

Destructor.

3.17.2.3 CellML_MathMLCiElement::CellML_MathMLCiElement () `[private]`

Constructor.

3.17.2.4 CellML_MathMLCiElement::CellML_MathMLCiElement (const CellML_MathMLCiElement * *other*) `[private]`

Copy constructor.

3.17.3 Member Function Documentation

3.17.3.1 const char* CellML_MathMLCiElement::getIdentifier () const

Get the identifier name.

Returns :
The identifier name.

3.17.3.2 float CellML_MathMLCiElement::getInitialValue () const

Get the initial value for this `ci` element.

Currently this field is only set when a CellML Model's mathematics is resolved into a single MathML Document by the **CellML_Equation::getAllMath**(`const CellML_Model&`) (p. 41) method.

Returns :
The value of the initial value of the **CellML_Variable** (p. 163) this `ci` element references.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if the initial value has not been set.

See also:
isInitialValueSet() (p. 80). , **CellML_Equation::getAllMath**(const CellML_Model&) (p. 41).

3.17.3.3 const char* CellML_MathMLCiElement::getType () const

Get the type.

Returns :
The type of the `ci` element.

3.17.3.4 bool CellML_MathMLCiElement::isInitialValueSet () const

Check if this `ci` element has had its initial value set.

Returns :
true if the initial value has been set; false otherwise.

3.17.3.5 void CellML_MathMLCiElement::setIdentifier (const char * *identifier*)

Set the identifier name.

Parameters:
identifier The new identifier name.

3.17.3.6 void CellML_MathMLCiElement::setInitialValue (float *value*)

Set the initial value for this `ci` element.

Currently this field is only set when a CellML Model's mathematics is resolved into a single MathML Document by the **CellML_Equation::getAllMath**(`const CellML_Model&`) (p. 41) method.

Parameters:

value The value of the initial value of the **CellML_Variable** (p. 163) this `ci` element references.

See also:

CellML_Equation::getAllMath(const CellML_Model&) (p. 41).

3.17.3.7 void CellML_MathMLCiElement::setType (const char * *type*)

Set the type.

Parameters:

type The type of the `ci` element. Values include, but are not restricted to, `integer`, `rational`, `real`, `float`, `complex`, `complex-polar`, `complex-cartesian`, `constant`, any of the MathML content container types (`vector`, `matrix`, `set`, `list` etc.) or their types.

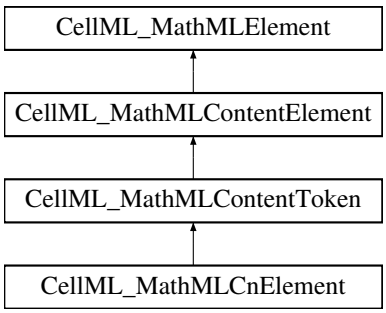
The documentation for this class was generated from the following file:

- CellML_MathMLCiElement.hpp

3.18 CellML_MathMLCnElement Class Reference

The `cn` element is used to specify actual numeric constants.

Inheritance diagram for CellML_MathMLCnElement::



Public Methods

- **CellML_MathMLCnElement** (const **CellML_MathMLMathElement** *ownerMathElement, float value)

  Constructor.

- ~**CellML_MathMLCnElement** ()

Destructor.

- const char * **getType** () const
  Get the type.
- void **setType** (const char *type)
  Set the type.
- const char * **getBase** () const
  Get the base.
- void **setBase** (const char *base)
  Set the base.
- int **getNargs** () const
  Get the number of arguments.
- const char * **getUnitsName** () const
  Get the CellML units name.
- void **setUnitsName** (const char *unitsName)
  Set the CellML units name.
- float **getValue** () const
  Get the value of the `cn` *element.*

Private Methods

- **CellML_MathMLCnElement** ()
  Constructor.
- **CellML_MathMLCnElement** (const CellML_MathMLCnElement *other)
  Copy constructor.

3.18.1 Detailed Description

The `cn` element is used to specify actual numeric constants.

This is extended to include the use of CellML units, which must be associated with all numeric constants.

3.18.2 Constructor & Destructor Documentation

3.18.2.1 CellML_MathMLCnElement::CellML_MathMLCnElement (const CellML_MathMLMathElement * *ownerMathElement*, float *value*)

Constructor.

3.18.2.2 CellML_MathMLCnElement::~CellML_MathMLCnElement ()

Destructor.

3.18.2.3 CellML_MathMLCnElement::CellML_MathMLCnElement () [private]

Constructor.

3.18.2.4 CellML_MathMLCnElement::CellML_MathMLCnElement (const CellML_MathMLCnElement * *other*) [private]

Copy constructor.

3.18.3 Member Function Documentation

3.18.3.1 const char* CellML_MathMLCnElement::getBase () const

Get the base.

Returns :
A string representing an integer between 2 and 36; the base of the numerical representation.

3.18.3.2 int CellML_MathMLCnElement::getNargs () const

Get the number of arguments.

Returns :
The number of `sep`-separated arguments for this `cn` element.

3.18.3.3 const char* CellML_MathMLCnElement::getType () const

Get the type.

Returns :
The type of the `cn` element.

3.18.3.4 const char* CellML_MathMLCnElement::getUnitsName () const

Get the CellML units name.

Returns :
A null-terminated string representation of the name of the CellML units associated with this constant.

3.18.3.5 float CellML_MathMLCnElement::getValue () const

Get the value of the `cn` element.

Since CellML allows only real values for `cn` elements, we can be a bit slack here ??

Returns :
The value of this `cn` element.

3.18.3.6 void CellML_MathMLCnElement::setBase (const char * *base*)

Set the base.

Parameters:

base A string representing an integer between 2 and 36; the base of the numerical representation.

3.18.3.7 void CellML_MathMLCnElement::setType (const char * *type*)

Set the type.

Parameters:

type The type of the `cn` element. Values include, but are not restricted to, `e-notation`, `integer`, `rational`, `real`, `float`, `complex`, `complex-polar`, `complex-cartesian`, and `constant`.

3.18.3.8 void CellML_MathMLCnElement::setUnitsName (const char * *unitsName*)

Set the CellML units name.

Parameters:

unitsName The name of the units to associate with this constant.

Exceptions:

CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the `unitsName` is an illegal CellML identifier.

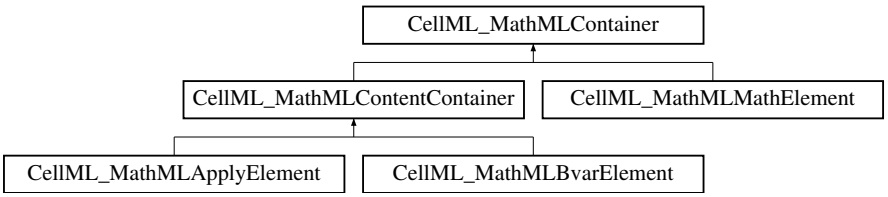
The documentation for this class was generated from the following file:

- CellML_MathMLCnElement.hpp

3.19 CellML_MathMLContainer Class Reference

This is an abstract interface containing functionality required by MathML elements that may contain arbitrarily many child elements.

Inheritance diagram for CellML_MathMLContainer::



Public Methods

- virtual int **getNArguments** () const=0

Get the number of arguments.

- virtual const **CellML_MathMLNodeList** * **getArguments** () const=0

  Get the argument list.

- virtual const **CellML_MathMLNodeList** * **getDeclarations** () const=0

  Get the declaration list.

- virtual const **CellML_MathMLElement** * **getArgument** (int index) const=0

  Get an argument.

- virtual **CellML_MathMLElement** * **setArgument** (**CellML_MathMLElement** *newArgument, int index)=0

  Set an argument.

- virtual **CellML_MathMLElement** * **insertArgument** (**CellML_MathMLElement** *newArgument, int index)=0

  Insert an argument.

- virtual void **deleteArgument** (int index)=0

  Delete an argument.

- virtual **CellML_MathMLElement** * **removeArgument** (int index)=0

  Remove an argument.

- virtual class CellML_MathMLDeclareElement * **getDeclaration** (int index) const=0

  Get a declaration.

- virtual class CellML_MathMLDeclareElement * **setDeclaration** (CellML_MathMLDeclareElement *newDeclaration, int index)=0

  Set a declaration.

- virtual CellML_MathMLDeclareElement * **insertDeclaration** (CellML_MathMLDeclareElement *newDeclaration, int index)=0

  Insert a declaration.

- virtual void **deleteDeclaration** (int index)=0

  Delete a declaration.

- virtual CellML_MathMLDeclareElement * **removeDeclaration** (int index)=0

  Remove a declaration.

3.19.1 Detailed Description

This is an abstract interface containing functionality required by MathML elements that may contain arbitrarily many child elements.

No elements are directly supported by this interface; all instances are instances of either `MathMLPresentationContainer`, `MathMLContentContainer`, or `MathMLMathElement`.

3.19.2 Member Function Documentation

3.19.2.1 virtual void CellML_MathMLContainer::deleteArgument (int *index*) [pure virtual]

Delete an argument.

This method deletes the `index`-th child element that is an argument of this element. Note that child elements which are qualifier elements or `declare` elements are not counted in determining the `index`-th argument.

Parameters:
index The one-based index of the argument to be deleted.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of child elements.

3.19.2.2 virtual void CellML_MathMLContainer::deleteDeclaration (int *index*) [pure virtual]

Delete a declaration.

This method deletes the `MathMLDeclareElement` representing the `index`-th `declare` child element of this element. Note that `index` is the position in the list of `declare` element children, as opposed to the position in the list of all child `Nodes`.

Parameters:
index The one-based index of the `declare` element to be removed.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is greater than the number of child `declare` elements.

3.19.2.3 virtual const CellML_MathMLElement* CellML_MathMLContainer::getArgument (int *index*) const [pure virtual]

Get an argument.

This method returns the `index`-th child argument element of this element. This frequently differs from the value of `Node::childNodes().item(index)`, as qualifier elements and `declare` elements are not counted.

Parameters:
index The one-based index of the argument to be retrieved.

Returns :
A `MathMLElement` representing the `index`-th argument of this element.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is greater than the number of child elements.

3.19.2.4 virtual const CellML_MathMLNodeList* CellML_MathMLContainer::getArguments () const [pure virtual]

Get the argument list.

Accesses the child `MathMLElements` of this element which are arguments of it, as a `MathMLNodeList`. Note that this list does not contain any `MathMLElements` representing qualifier elements or `declare` elements.

Returns :
The list of arguments.

3.19.2.5 virtual class CellML_MathMLDeclareElement* CellML_MathMLContainer::getDeclaration (int *index*) const [pure virtual]

Get a declaration.

This method retrieves the `index`-th child `declare` element of this element.

Parameters:
index A one-based index into the list of child `declare` elements of this element giving the position of the `declare` element to be retrieved.

Returns :
The `MathMLDeclareElement` representing the `index`-th child `declare`.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if `index` is greater than the number of child `declare` elements.

3.19.2.6 virtual const CellML_MathMLNodeList* CellML_MathMLContainer::getDeclarations () const [pure virtual]

Get the declaration list.

Provides access to the `declare` elements which are children of this element, in a `MathMLNodeList`. All elements in this list must be `MathMLDeclareElements`.

Returns :
The list of declarations.

3.19.2.7 virtual int CellML_MathMLContainer::getNArguments () const [pure virtual]

Get the number of arguments.

The number of child elements of this element which represent arguments of the element, as opposed to qualifiers or `declare` elements. Thus for a `MathMLContentContainer` it does not contain elements representing bound variables, conditions, separators, degrees, or upper or lower limits (`bvar, condition, sep, degree, lowlimit, or uplimit`).

`Returns :`
`The number of arguments.`

3.19.2.8 virtual CellML_MathMLElement* CellML_MathMLContainer::insertArgument (CellML_MathMLElement * *newArgument*, int *index*) [pure virtual]

Insert an argument.

This method inserts newArgument before the current index-th argument of this element. If index is 0, or if index is one more than the current number of arguments, newArgument is appended as the last argument. This frequently differs from setting the node at Node::childNodes().item(index), as qualifier elements and declare elements are not counted.

Parameters:

newArgument A MathMLElement representing the element that is to be inserted as a child argument of this element.

index The one-based index of the position before which newArgument is to be inserted. The first argument is numbered 1.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than one more than the number of child arguments.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child argument of the type of newArgument, or, for MathMLContentContainers, if newArgument represents a qualifier element.

3.19.2.9 virtual CellML_MathMLDeclareElement* CellML_MathMLContainer::insertDeclaration (CellML_MathMLDeclareElement * *newDeclaration*, int *index*) [pure virtual]

Insert a declaration.

This method inserts newDeclaration before the current index-th child declare element of this element. If index is 0, newDeclaration is appended as the last child declare element.

Parameters:

newDeclaration A MathMLDeclareElement to be inserted as the index-th child declare element.

index A one-based index into the list of child declare elements of this element giving the position before which newDeclaration is to be inserted. If index is 0 or if it is one more than the number of child declare children, newDeclaration is appended as the last child declare element.

Returns :

The MathMLDeclareElement child of this element representing newDeclaration in the DOM.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than one more than the number of child declare elements.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit child declare elements.

3.19.2.10 virtual CellML_MathMLElement* CellML_MathMLContainer::removeArgument (int *index*) [pure virtual]

Remove an argument.

This method deletes the index-th child element that is an argument of this element, and returns it to the caller. Note that child elements that are qualifier elements or declare elements are not counted in determining the index-th argument.

Parameters:
index The one-based index of the argument to be removed.

Returns :
A MathMLElement representing the argument being removed.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of child elements.

3.19.2.11 virtual CellML_MathMLDeclareElement* CellML_MathMLContainer::removeDeclaration (int *index*) [pure virtual]

Remove a declaration.

This method removes the MathMLDeclareElement representing the index-th declare child element of this element, and returns it to the caller. Note that index is the position in the list of declare element children, as opposed to the position in the list of all child Nodes.

Parameters:
index The one-based index of the declare element to be removed.

Returns :
The MathMLDeclareElement being removed as a child Node of this element.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of child declare elements.

3.19.2.12 virtual CellML_MathMLElement* CellML_MathMLContainer::setArgument (CellML_MathMLElement * *newArgument*, int *index*) [pure virtual]

Set an argument.

This method sets newArgument as the index-th argument of this element. If there is currently an index-th argument, it is replaced by newArgument. This frequently differs from setting the node at Node::childNodes().item(index), as qualifier elements and declare elements are not counted.

Parameters:
newArgument A MathMLElement representing the element that is to be set as the index-th argument of this element.

index The index of the argument that is to be set to newArgument. The first argument is numbered 1. If index is one more than the current number of arguments, a new argument is appended.

Returns :
The MathMLElement child of this element that represents the new argument in the DOM.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than one more than the number of child elements.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child argument of the type of newArgument, or, for MathMLContentContainers, if newArgument represents a qualifier element.

3.19.2.13 virtual class CellML_MathMLDeclareElement* CellML_MathMLContainer::setDeclaration (CellML_MathMLDeclareElement * *newDeclaration*, int *index*) [pure virtual]

Set a declaration.

This method inserts newDeclaration as the index-th child declaration of this element. If there is already an index-th declare child element, it is replaced by newDeclaration.

Parameters:
newDeclaration A MathMLDeclareElement to be inserted as the index-th child declare element.

index A one-based index into the list of child declare elements of this element giving the position into which newDeclaration is to be inserted. If index is one more than the number of declare children of this element, newDeclaration is appended as the last declare child.

Returns :
The MathMLDeclareElement being inserted.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than one more than the number of child declare elements.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit child declare elements.

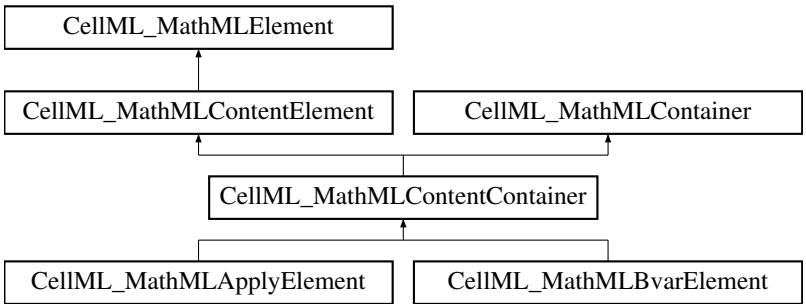
The documentation for this class was generated from the following file:

- CellML_MathMLContainer.hpp

3.20 CellML_MathMLContentContainer Class Reference

This interface supports the MathML Content elements that may contain child Content elements.

Inheritance diagram for CellML_MathMLContentContainer::



Public Methods

- virtual int **getNBoundVariables** () const=0

  Get number of bound variables.

- virtual const **CellML_MathMLNodeList** * **getBoundVariables** () const=0

  Get the list of bound variables.

- virtual class CellML_MathMLConditionElement * **getCondition** () const=0

  Get the condition element.

- virtual void **setCondition** (class CellML_MathMLConditionElement *condition)=0

  Set the condition element.

- virtual **CellML_MathMLElement** * **getOpDegree** () const=0

  Get the degree element.

- virtual void **setOpDegree** (**CellML_MathMLElement** *opDegree)=0

  Set the degree element.

- virtual **CellML_MathMLElement** * **getDomainOfApplication** () const=0

  Get the domain of application element.

- virtual void **setDomainOfApplication** (**CellML_MathMLElement** *domainOfApplication)=0

  Set the domain of application element.

- virtual const **CellML_MathMLElement** * **getMomentAbout** () const=0

  Get the moment about element.

- virtual void **setMomentAbout** (**CellML_MathMLElement** *momentAbout)=0

  Set the moment about element.

- virtual class **CellML_MathMLBvarElement** * **getBoundVariable** (int index) const=0

  Get a bound variable.

- virtual class **CellML_MathMLBvarElement** * **insertBoundVariable** (class **CellML_MathMLBvarElement** *newBVar, int index)=0

  Insert a bound variable.

- virtual class **CellML_MathMLBvarElement** * **setBoundVariable** (class **CellML_MathMLBvarElement** *newBVar, int index)=0

  Set a bound variable.

- virtual void **deleteBoundVariable** (int index)=0

  Delate a bound variable.

- virtual class **CellML_MathMLBvarElement** * **removeBoundVariable** (int index)=0

  Remove a bound variable.

3.20.1 Detailed Description

This interface supports the MathML Content elements that may contain child Content elements.

The elements directly supported by MathMLContentContainer include: reln (deprecated), lambda, lowlimit, uplimit, degree, domainofapplication, and momentabout. Interfaces derived from MathMLContentContainer support the elements apply, fn, interval, condition, declare, bvar, set, list, vector, matrix, and matrixrow.

3.20.2 Member Function Documentation

3.20.2.1 virtual void CellML_MathMLContentContainer::deleteBoundVariable (int *index*) [pure virtual]

Delate a bound variable.

This method deletes the index-th MathMLBvarElement child of the MathMLElement. This has the effect of removing this bound variable from the list of qualifiers affecting the element this represents.

Parameters:

index The one-based index into the bound variable children of this element of the MathMLBvarElement to be removed.

3.20.2.2 virtual class CellML_MathMLBvarElement* CellML_MathMLContentContainer::getBoundVariable (int *index*) const [pure virtual]

Get a bound variable.

This method retrieves the index-th MathMLBvarElement child of the MathMLElement. Note that only bvar child elements are counted in determining the index-th bound variable.

Parameters:

index The one-based index into the bound variable children of this element of the MathMLBvarElement to be retrieved.

Returns :
The MathMLBvarElement representing the index-th bvar child of this element.

3.20.2.3 virtual const CellML_MathMLNodeList* CellML_MathMLContentContainer::getBoundVariables () const [pure virtual]

Get the list of bound variables.

Returns :
The bound variable list of this node.

3.20.2.4 virtual class CellML_MathMLConditionElement* CellML_MathMLContentContainer::getCondition () const [pure virtual]

Get the condition element.

Returns :
The condition child element of this node.

3.20.2.5 virtual CellML_MathMLElement* CellML_MathMLContentContainer::getDomainOfApplication () const [pure virtual]

Get the domain of application element.

Returns :
The domainofapplication child element of this node, if present. This may express, for instance, the domain of integration if this element is an apply element whose first child is an integral operator (int).

3.20.2.6 virtual const CellML_MathMLElement* CellML_MathMLContentContainer::getMomentAbout () const [pure virtual]

Get the moment about element.

Returns :
The momentabout child element of this node, if present. This typically expresses the point about which a statistical moment is to be calculated, if this element is an apply element whose first child is a moment.

3.20.2.7 virtual int CellML_MathMLContentContainer::getNBoundVariables () const [pure virtual]

Get number of bound variables.

Returns :
The number of bvar child elements of this element.

3.20.2.8 virtual CellML_MathMLElement* CellML_MathMLContentContainer::getOpDegree () const [pure virtual]

Get the degree element.

Returns :
The degree child element of this node. This expresses, for instance, the degree of differentiation if this element is a bvar child of an apply element whose first child is a diff or partialdiff. If this is an apply element whose first child is a partialdiff, the opDegree attribute, if present, represents the total degree of differentiation.

3.20.2.9 virtual class CellML_MathMLBvarElement* CellML_MathMLContentContainer::insertBoundVariable (class CellML_MathMLBvarElement * *newBVar*, int *index*) [pure virtual]

Insert a bound variable.

This method inserts a MathMLBvarElement as a child node before the current index-th bound variable child of this MathMLElement. If index is 0, newBVar is appended as the last bound variable child. This has the effect of adding a bound variable to the expression this element represents. Note that the new bound variable is inserted as the index-th bvar child node, not necessarily as the index-th child node. The point of the method is to allow insertion of bound variables without requiring the caller to calculate the exact order of child qualifier elements.

Parameters:
newBVar A MathMLBvarElement representing the bvar element being added.
index The one-based index into the bound variable children of this element before which newBVar is to be inserted.

Returns :
The MathMLBvarElement being added.

Exceptions:
CellML_Exception (p. 41) : INVALID_MODIFICATION_ERR: Raised if this element does not permit child bvar elements.

3.20.2.10 virtual class CellML_MathMLBvarElement* CellML_MathMLContentContainer::removeBoundVariable (int *index*) [pure virtual]

Remove a bound variable.

This method removes the index-th MathMLBvarElement child of the MathMLElement and returns it to the caller. This has the effect of removing this bound variable from the list of qualifiers affecting the element this represents.

Parameters:
index The one-based index into the bound variable children of this element of the MathMLBvarElement to be removed.

Returns :
The MathMLBvarElement being removed.

3.20.2.11 virtual class CellML_MathMLBvarElement* CellML_MathMLContentContainer::setBoundVariable (class CellML_MathMLBvarElement * *newBVar*, int *index*) [pure virtual]

Set a bound variable.

This method sets the index-th bound variable child of this MathMLElement to newBVar. This has the effect of setting a bound variable in the expression this element represents. Note that the new bound variable is inserted as the index-th bvar child node, not necessarily as the index-th child node. The point of the method is to allow insertion of bound variables without requiring the caller to calculate the exact order of child qualifier elements. If there is already a bvar at the index-th position, it is replaced by newBVar.

Parameters:

newBVar The new MathMLBvarElement child of this element being set.

index The one-based index into the bound variable children of this element at which newBVar is to be inserted.

Returns :

The MathMLBvarElement being set.

Exceptions:

CellML_Exception (p. 41) : INVALID_MODIFICATION_ERR: Raised if this element does not permit child bvar elements.

3.20.2.12 virtual void CellML_MathMLContentContainer::setCondition (class CellML_MathMLConditionElement * *condition*) [pure virtual]

Set the condition element.

Parameters:

condition The condition child element of this node.

Exceptions:

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child condition element. In particular, raised if this element is not a apply, set, or list.

3.20.2.13 virtual void CellML_MathMLContentContainer::setDomainOfApplication (CellML_MathMLElement * *domainOfApplication*) [pure virtual]

Set the domain of application element.

Parameters:

domainOfApplication The domainofapplication child element of this node. This may express, for instance, the domain of integration if this element is an apply element whose first child is an integral operator (int).

Exceptions:

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child domainofapplication element.

3.20.2.14 virtual void CellML_MathMLContentContainer::setMomentAbout (CellML_MathMLElement * *momentAbout*) [pure virtual]

Set the moment about element.

Parameters:

momentAbout The momentabout child element of this node. This typically expresses the point about which a statistical moment is to be calculated, if this element is an apply element whose first child is a moment.

Exceptions:

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child momentabout element. In particular, raised if this element is not an apply whose first child is a moment.

3.20.2.15 virtual void CellML_MathMLContentContainer::setOpDegree (CellML_MathMLElement * *opDegree*) [pure virtual]

Set the degree element.

Parameters:

opDegree The degree child element of this node. This expresses, for instance, the degree of differentiation if this element is a bvar child of an apply element whose first child is a diff or partialdiff. If this is an apply element whose first child is a partialdiff, the opDegree attribute, if present, represents the total degree of differentiation.

Exceptions:

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if this element does not permit a child degree element. In particular, raised if this element is not a bvar or apply.

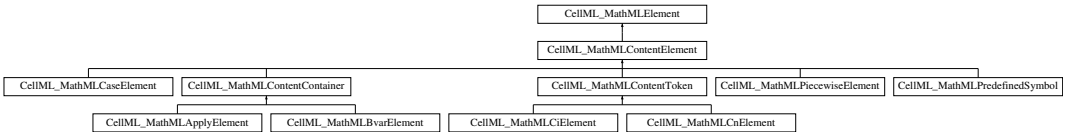
The documentation for this class was generated from the following file:

- CellML_MathMLContentContainer.hpp

3.21 CellML_MathMLContentElement Class Reference

This class is provided to serve as the base class for the content MathML elements.

Inheritance diagram for CellML_MathMLContentElement::



3.21.1 Detailed Description

This class is provided to serve as the base class for the content MathML elements.

Used in the MathML DOM to distinguish between content and presentation.

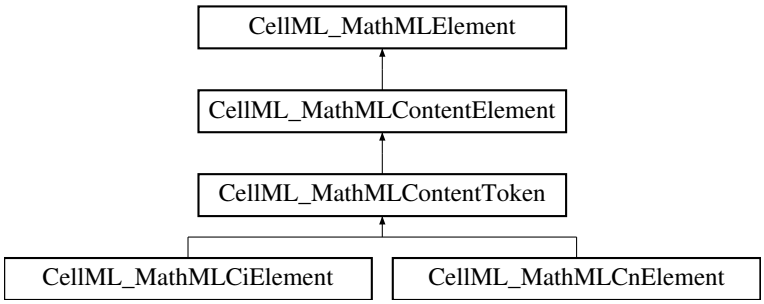
The documentation for this class was generated from the following file:

- CellML_MathMLContentElement.hpp

3.22 CellML_MathMLContentToken Class Reference

The base class from which the MathML content token elements (ci, cn, and csymbol) are derived.

Inheritance diagram for CellML_MathMLContentToken::



Public Methods

- virtual **CellML_MathMLNodeList** * **getArguments** () const=0

  Get the argument list.

- virtual DOMNode * **getArgument** (int index) const=0

  Get an argument.

- virtual DOMNode * **insertArgument** (DOMNode *newArgument, int index)=0

  Insert an argument.

- virtual DOMNode * **setArgument** (DOMNode *newArgument, int index)=0

  Set an argument.

- virtual void **deleteArgument** (int index)=0

  Delete an argument.

- virtual DOMNode * **removeArgument** (int index)=0

  Remove an argument.

3.22.1 Detailed Description

The base class from which the MathML content token elements (ci, cn, and csymbol) are derived.

3.22.2 Member Function Documentation

3.22.2.1 virtual void CellML_MathMLContentToken::deleteArgument (int *index*) [pure virtual]

Delete an argument.

A convenience method to delete the argument child located at the position referenced by index.

Parameters:
index Position of the argument to be deleted from the list of arguments. The first argument is numbered 1.

3.22.2.2 virtual DOMNode* CellML_MathMLContentToken::getArgument (int *index*) const [pure virtual]

Get an argument.

A convenience method to retrieve the child argument at the position referenced by index. Note that this is not necessarily the same as the index-th child Node of this Element; in particular, sep elements will not be counted.

Parameters:
index Position of desired argument in the list of arguments. The first argument is numbered 1.

Returns :
The Node retrieved.

3.22.2.3 virtual CellML_MathMLNodeList* CellML_MathMLContentToken::getArguments () const [pure virtual]

Get the argument list.

The arguments of this element, returned as a MathMLNodeList. Note that this is not necessarily the same as Node::childNodes, particularly in the case of the cn element. The reason is that the sep elements that are used to separate the arguments of a cn are not returned.

Returns :
The arguments of this element.

3.22.2.4 virtual DOMNode* CellML_MathMLContentToken::insertArgument (DOMNode * *newArgument*, int *index*) [pure virtual]

Insert an argument.

A convenience method to insert newArgument before the current index-th argument child of this element. If index is 0, newArgument is appended as the last argument.

Parameters:

newArgument Argument to be inserted as the index-th argument.

index Position before which newArgument is to be inserted. The first argument is numbered 1.

Returns :

The Node inserted. This is the element within the DOM.

3.22.2.5 virtual DOMNode* CellML_MathMLContentToken::removeArgument (int *index*) [pure virtual]

Remove an argument.

A convenience method to delete the argument child located at the position referenced by index, and to return it to the caller.

Parameters:

index Position of the argument to be deleted from the list of arguments. The first argument is numbered 1.

Returns :

A Node representing the deleted argument.

3.22.2.6 virtual DOMNode* CellML_MathMLContentToken::setArgument (DOMNode * *newArgument*, int *index*) [pure virtual]

Set an argument.

A convenience method to set an argument child at the position referenced by index. If there is currently an argument at this position, it is replaced by newArgument.

Parameters:

newArgument Argument to be inserted as the index-th argument.

index Position of the argument that is to be set to newArgument in the list of arguments. The first argument is numbered 1.

Returns :

The Node inserted. This is the element within the DOM.

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

- CellML_MathMLContentToken.hpp

3.23 CellML_MathMLDocument Class Reference

This interface extends the Document interface to add access to document properties relating to navigation.

Public Methods

- **CellML_MathMLDocument** ()
  Create a MathML document.
- **CellML_MathMLDocument** (const CellML_MathMLDocument *other)
  Copy constructor.
- **~CellML_MathMLDocument** ()
  Destructor.
- const char * **getReferrer** () const
  The URI of the page that linked to this document, if available.
- const char * **getDomain** () const
  The domain name of the server that served the document.
- const char * **getURI** () const
  The complete URI of this document.
- void **importNode** (const DOMNode *node)
  Import a DOM node into the math *element of this document.*
- **CellML_MathMLApplyElement** * **createApplyElement** () const
  Create an apply *element.*

3.23.1 Detailed Description

This interface extends the Document interface to add access to document properties relating to navigation.

The documentElement attribute for a MathMLDocument should be the MathMLMathElement representing the top-level math element which is the root of the document.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 CellML_MathMLDocument::CellML_MathMLDocument ()

Create a MathML document.

Creates a MathMLDocument with a minimal tree containing only a MathMLMathElement corresponding to a MathML math element. The MathMLMathElement is empty, having no child elements or non-default attributes; it is the root element of the document, and is the element accessed via the documentElement attribute of the MathMLDocument. Note that a MathMLDocument object should only be created for a stand-alone MathML document.

3.23.2.2 CellML_MathMLDocument::CellML_MathMLDocument (const CellML_MathMLDocument * *other*)

Copy constructor.

3.23.2.3 CellML_MathMLDocument::~CellML_MathMLDocument ()

Destructor.

3.23.3 Member Function Documentation

3.23.3.1 class CellML_MathMLApplyElement* CellML_MathMLDocument::createApplyElement () const

Create an apply element.

3.23.3.2 const char* CellML_MathMLDocument::getDomain () const

The domain name of the server that served the document.

The domain name of the server that served the document, or null if the server cannot be identified by a domain name, or if it is not available. If this is not a stand-alone MathML document (e.g. is embedded in an XHTML document), this may be retrieved from the parent Document if available.

3.23.3.3 const char* CellML_MathMLDocument::getReferrer () const

The URI of the page that linked to this document, if available.

This is null if the user navigated directly to the page. If this is not a stand-alone MathML document (e.g. is embedded in an XHTML document), this may be retrieved from the parent Document if available.

3.23.3.4 const char* CellML_MathMLDocument::getURI () const

The complete URI of this document.

The complete URI of this document. This is null if this is not a stand-alone MathML document.

3.23.3.5 void CellML_MathMLDocument::importNode (const DOMNode * *node*)

Import a DOM node into the math element of this document.

Used to create a MathML representation from a top-level child node of a math element.

Parameters:

node The DOM node to import.

Exceptions:

CellML_Exception (p. 41) NOT_IMPLEMENTED_ERR: Raised when the given node can not be imported - due to only having a subset of MathML 2.0 implemented.

CellML_Exception (p. 41) INVALID_MATHML_ERR: Raised when trying to import invalid MathML into the document.

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

- CellML_MathMLDocument.hpp

3.24 CellML_MathMLDocumentList Class Reference

This class is used to store lists of MathML documents, since each CellML component can have multiple math elements, each of which will form the basis of a MathML document in the component's list of documents.

Public Methods

Methods.

- **CellML_MathMLDocumentList** ()
  Constructor.
- **CellML_MathMLDocumentList** (const CellML_MathMLDocumentList *other)
  Copy constructor.
- **~CellML_MathMLDocumentList** ()
  Destructor.
- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- const **CellML_MathMLDocument** * **get** (const int index) const
  Get a document from the list.
- **CellML_MathMLDocument** * **get** (const int index)
  Get a document from the list.
- void **append** (**CellML_MathMLDocument** *newDocument)
  Append a document to the list.
- **CellML_MathMLDocument** * **remove** (int index)
  Remove the given document from the list.

3.24.1 Detailed Description

This class is used to store lists of MathML documents, since each CellML component can have multiple math elements, each of which will form the basis of a MathML document in the component's list of documents.

3.24.2 Constructor & Destructor Documentation

3.24.2.1 CellML_MathMLDocumentList::CellML_MathMLDocumentList ()

Constructor.

Create an empty list.

3.24.2.2 CellML_MathMLDocumentList::CellML_MathMLDocumentList (const CellML_MathMLDocumentList * *other*)

Copy constructor.

Create a new list initialised with the contents of the other list.

3.24.2.3 CellML_MathMLDocumentList::~CellML_MathMLDocumentList ()

Destructor.

Releases all the documents in the list.

3.24.3 Member Function Documentation

3.24.3.1 void CellML_MathMLDocumentList::append (CellML_MathMLDocument * *newDocument*)

Append a document to the list.

Adds newDocument to the end of the list.

Parameters:
newDocument The document object to add to the end of this list.

3.24.3.2 CellML_MathMLDocument* CellML_MathMLDocumentList::get (const int *index*)

Get a document from the list.

Used to get a document object from this list by its position in the list.

Parameters:
index The index of the desired document (valid range is from 0 to **length()** (p. 104)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the valid range for this document list.

See also:
length() (p. 104)

3.24.3.3 const CellML_MathMLDocument* CellML_MathMLDocumentList::get (const int *index*) const

Get a document from the list.

Used to get a document object from this list by its position in the list.

Parameters:
index The index of the desired document (valid range is from 0 to **length()** (p. 104)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the valid range for this document list.

See also:
length() (p. 104)

3.24.3.4 bool CellML_MathMLDocumentList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.24.3.5 int CellML_MathMLDocumentList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.24.3.6 CellML_MathMLDocument* CellML_MathMLDocumentList::remove (int *index*)

Remove the given document from the list.

Removes the document at the index-th position in the list and returns it.

Parameters:
index The index of the document to be removed.

Returns :
The document removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the range 0 to **length()** (p. 104)-1

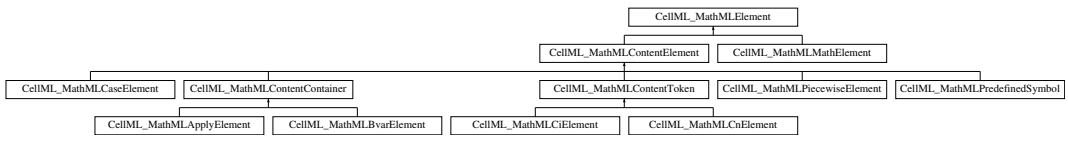
The documentation for this class was generated from the following file:

- CellML_MathMLDocumentList.hpp

3.25 CellML_MathMLElement Class Reference

All CellML MathML objects derive from this class, except lists? This is a custom implementation of the W3C Document Object Model for MathML (??).

Inheritance diagram for CellML_MathMLElement::



Public Types

- enum **ElementType**

  Element types.

Public Methods

- virtual const **CellML_MathMLMathElement** * **getOwnerMathElement** () const=0

  Get the parent MathML math element.

- virtual int **getElementType** () const=0

  Get the type of the element.

- virtual CellML_MathMLElement * **cloneElement** (bool deep) const=0

  Return a clone of the element.

3.25.1 Detailed Description

All CellML MathML objects derive from this class, except lists? This is a custom implementation of the W3C Document Object Model for MathML (??).

3.25.2 Member Enumeration Documentation

3.25.2.1 enum CellML_MathMLElement::ElementType

Element types.

3.25.3 Member Function Documentation

3.25.3.1 virtual CellML_MathMLElement* CellML_MathMLElement::cloneElement (bool *deep*) const [pure virtual]

Return a clone of the element.

3.25.3.2 virtual int CellML_MathMLElement::getElementType () const [pure virtual]

Get the type of the element.

Returns :
    This element's actual type.

3.25.3.3 virtual const CellML_MathMLMathElement* CellML_MathMLElement::getOwnerMathElement () const [pure virtual]

Get the parent MathML math element.

Returns :
The parent MathML element of this MathML element.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised when this is a top-level math element with no parent MathML element.

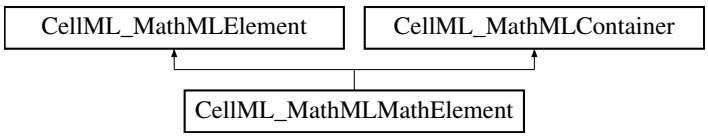
The documentation for this class was generated from the following file:

- CellML_MathMLElement.hpp

3.26 CellML_MathMLMathElement Class Reference

This interface represents the top-level MathML math element.

Inheritance diagram for CellML_MathMLMathElement::



Public Methods

- **CellML_MathMLMathElement** (const **CellML_MathMLDocument** *ownerDocument)
  Constructor.
- **CellML_MathMLMathElement** (const CellML_MathMLMathElement *other)
  Copy constructor.
- ~**CellML_MathMLMathElement** ()
  Destructor.
- const **CellML_MathMLDocument** * **getOwnerDocument** () const
  Get the owner MathML Document.

3.26.1 Detailed Description

This interface represents the top-level MathML math element.

3.26.2 Constructor & Destructor Documentation

3.26.2.1 CellML_MathMLMathElement::CellML_MathMLMathElement (const CellML_MathMLDocument * *ownerDocument*)

Constructor.

3.26.2.2 CellML_MathMLMathElement::CellML_MathMLMathElement (const CellML_MathMLMathElement * *other*)

Copy constructor.

3.26.2.3 CellML_MathMLMathElement::~CellML_MathMLMathElement ()

Destructor.

3.26.3 Member Function Documentation

3.26.3.1 const CellML_MathMLDocument* CellML_MathMLMathElement::getOwnerDocument () const

Get the owner MathML Document.

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

- CellML_MathMLMathElement.hpp

3.27 CellML_MathMLNodeList Class Reference

This interface is provided as a specialization of the NodeList interface.

Public Methods

- **CellML_MathMLNodeList** ()
  Constructor.
- **CellML_MathMLNodeList** (const CellML_MathMLNodeList *other)
  Copy constructor.
- **~CellML_MathMLNodeList** ()
  Destructor.
- int **getLength** () const
  Get the current length of the list.
- **CellML_MathMLElement** * **item** (int index) const
  Get an item from the list.
- void **append** (**CellML_MathMLElement** *newElement)
  Append a element to the list.
- **CellML_MathMLElement** * **remove** (int index)
  Remove the given element from the list.

3.27.1 Detailed Description

This interface is provided as a specialization of the NodeList interface.

The child Nodes of this NodeList must be MathMLElements or Text nodes.

Note that MathMLNodeLists are frequently used in the DOM as values of readonly attributes, encapsulating, for instance, various collections of child elements. When used in this way, these objects are always understood to be live, in the sense that changes to the document are immediately reflected in them.

3.27.2 Constructor & Destructor Documentation

3.27.2.1 CellML_MathMLNodeList::CellML_MathMLNodeList ()

Constructor.

3.27.2.2 CellML_MathMLNodeList::CellML_MathMLNodeList (const CellML_MathMLNodeList * *other*)

Copy constructor.

3.27.2.3 CellML_MathMLNodeList::~CellML_MathMLNodeList ()

Destructor.

3.27.3 Member Function Documentation

3.27.3.1 void CellML_MathMLNodeList::append (CellML_MathMLElement * *newElement*)

Append a element to the list.

Adds newElement to the end of the list.

Parameters:
newElement The element object to add to the end of this list.

3.27.3.2 int CellML_MathMLNodeList::getLength () const

Get the current length of the list.

Returns :
The current number of items in the list.

3.27.3.3 CellML_MathMLElement* CellML_MathMLNodeList::item (int *index*) const

Get an item from the list.

Parameters:
index The index of the desired item from the list.

Returns :
The index-th element in the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the valid range for this list (0 to **getLength**() (p. 108)-1).

3.27.3.4 CellML_MathMLElement* CellML_MathMLNodeList::remove (int *index*)

Remove the given element from the list.

Removes the element at the index-th position in the list and returns it.

Parameters:
index The index of the element to be removed.

Returns :
The element removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the range 0 to length()-1

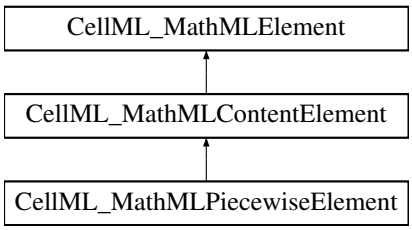
The documentation for this class was generated from the following file:

- CellML_MathMLNodeList.hpp

3.28 CellML_MathMLPiecewiseElement Class Reference

The piecewise element represents the piecewise definition of a function.

Inheritance diagram for CellML_MathMLPiecewiseElement::



Public Methods

- **CellML_MathMLPiecewiseElement** (const **CellML_MathMLMathElement** *ownerMathElement)

  Constructor.

- ~**CellML_MathMLPiecewiseElement** ()

  Destructor.

- int **getNPieces** () const

  Get the number of pieces.

- **CellML_MathMLNodeList** * **getPieces** () const

  Get the list of pieces.

- **CellML_MathMLContentElement** * **getOtherwise** () const

  Get the otherwise element.

- void **setOtherwise** (**CellML_MathMLContentElement** *otherwise)

  Set the otherwise element.

- const class **CellML_MathMLCaseElement** * **getCase** (int index) const

  Get a case element.

- **CellML_MathMLCaseElement** * **setCase** (int index, class **CellML_MathMLCaseElement** *newCase)

  Set a case element.

- void **deleteCase** (int index)

  Delete a case element.

- **CellML_MathMLCaseElement** * **removeCase** (int index)

  Remove a case element.

- **CellML_MathMLCaseElement** * **insertCase** (**CellML_MathMLCaseElement** *newCase, int index)

  Insert a case element.

- **CellML_MathMLContentElement** * **getCaseValue** (int index) const

  Get a case value element.

- **CellML_MathMLContentElement** * **setCaseValue** (int index, **CellML_MathMLContentElement** *value)

  Set a case value element.

- **CellML_MathMLContentElement** * **getCaseCondition** (int index) const

  Get a case condition element.

- **CellML_MathMLContentElement** * **setCaseCondition** (int index, **CellML_MathMLContentElement** *condition)

  Set a case condition element.

Private Methods

- **CellML_MathMLPiecewiseElement** ()

  Constructor.

- **CellML_MathMLPiecewiseElement** (const CellML_MathMLPiecewiseElement *other)

  Copy constructor.

3.28.1 Detailed Description

The piecewise element represents the piecewise definition of a function.

It contains child piece elements, each represented by a MathMLCaseElement, giving the various conditions and associated function value specifications in the function definition, and an optional otherwise child element, represented by a MathMLContentElement, giving the 'default' value of the function; that is, the value to be assigned when none of the conditions specified in the piece child elements hold.

3.28.2 Constructor & Destructor Documentation

3.28.2.1 CellML_MathMLPiecewiseElement::CellML_MathMLPiecewiseElement (const CellML_MathMLMathElement * *ownerMathElement*)

Constructor.

3.28.2.2 CellML_MathMLPiecewiseElement::~CellML_MathMLPiecewiseElement ()

Destructor.

3.28.2.3 CellML_MathMLPiecewiseElement::CellML_MathMLPiecewiseElement () [private]

Constructor.

3.28.2.4 CellML_MathMLPiecewiseElement::CellML_MathMLPiecewiseElement (const CellML_MathMLPiecewiseElement * *other*) [private]

Copy constructor.

3.28.3 Member Function Documentation

3.28.3.1 void CellML_MathMLPiecewiseElement::deleteCase (int *index*)

Delete a case element.

A convenience method to delete the child piece at the position referenced by index. The deletion changes the indices of the following pieces.

Parameters:

index Position of the piece to be deleted. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of pieces in this element.

3.28.3.2 const class CellML_MathMLCaseElement* CellML_MathMLPiecewiseElement::getCase (int *index*) const

Get a case element.

A convenience method to retrieve the child piece at the position referenced by index.

Parameters:

index Position of desired case in the list of cases. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position. If index is greater than the number of pieces, a null MathMLCaseElement is returned; no error is generated.

Returns :

The MathMLCaseElement retrieved.

3.28.3.3 CellML_MathMLContentElement* CellML_MathMLPiecewiseElement::getCaseCondition (int *index*) const

Get a case condition element.

A convenience method to retrieve the child of the piece at the position referenced by index which gives the condition for this case.

Parameters:

index Position of the piece whose condition is being requested in the list of pieces. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position.

Returns :

The MathMLContentElement representing the condition to be satisfied for the index-th case.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of pieces in this element.

3.28.3.4 CellML_MathMLContentElement* CellML_MathMLPiecewiseElement::getCaseValue (int *index*) const

Get a case value element.

A convenience method to retrieve the child of the index-th piece in this element which specifies the function value for that case.

Parameters:

index Position of the piece whose value is being requested in the list of pieces. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position.

Returns :

The MathMLContentElement representing the value to be taken by the function in the index-th case.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of pieces in this element.

3.28.3.5 int CellML_MathMLPiecewiseElement::getNPieces () const

Get the number of pieces.

Returns :
The number of piece children in this piecewise element.

3.28.3.6 CellML_MathMLContentElement* CellML_MathMLPiecewiseElement::getOtherwise () const

Get the otherwise element.

Returns :
A MathMLContentElement representing the value to be taken by the piecewise function when none of the conditions described in the piece children is true.

3.28.3.7 CellML_MathMLNodeList* CellML_MathMLPiecewiseElement::getPieces () const

Get the list of pieces.

Returns :
A MathMLNodeList containing one MathMLCaseElement representing each of the piece element children of this MathMLPiecewiseElement. The otherwise child (if present) is not contained in this MathMLNodeList.

3.28.3.8 CellML_MathMLCaseElement* CellML_MathMLPiecewiseElement::insertCase (CellML_MathMLCaseElement * *newCase*, int *index*)

Insert a case element.

A convenience method to insert a new piece child into this element.

Parameters:
index Position before which newCase is to be inserted. If index is 0, newCase is appended as the last piece child of this element. The otherwise child (if present) is not counted, regardless of its position.

newCase A MathMLCaseElement representing the piece to be inserted.

Returns :
The new MathMLCaseElement inserted. This is the actual Element in the DOM.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than one more than the number of pieces in this element.

3.28.3.9 class CellML_MathMLCaseElement* CellML_MathMLPiecewiseElement::removeCase (int *index*)

Remove a case element.

A convenience method to remove the child piece at the position referenced by index and return it to the caller. The removal changes the indices of the following pieces.

Parameters:

index Position of the piece to be removed. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position.

Returns :

The MathMLCaseElement removed.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of pieces in this element.

3.28.3.10 class CellML_MathMLCaseElement* CellML_MathMLPiecewiseElement::setCase (int *index*, class CellML_MathMLCaseElement * *newCase*)

Set a case element.

A convenience method to set the value of the child piece at the position referenced by index to the value of newCase.

Parameters:

index Position of the piece to be set to newCase. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position. If there is currently a piece at this position, it will be replaced by newCase. If index is one more than the number of piece child elements, a new one will be appended.

newCase A MathMLCaseElement representing the new value of the index-th piece child.

Returns :

The new MathMLCaseElement created.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than one more than the number of pieces in this element.

3.28.3.11 CellML_MathMLContentElement* CellML_MathMLPiecewiseElement::setCaseCondition (int *index*, CellML_MathMLContentElement * *condition*)

Set a case condition element.

A convenience method to set the condition for the index-th piece in this element.

Parameters:

index Position of the piece whose condition is being set in the list of pieces. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position.

condition A MathMLContentElement representing the condition to be associated to the index-th case.

Returns :

The MathMLContentElement which is inserted as the condition child of the index-th piece.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of pieces in this element.

3.28.3.12 CellML_MathMLContentElement* CellML_MathMLPiecewiseElement::setCaseValue (int *index*, CellML_MathMLContentElement * *value*)

Set a case value element.

A convenience method to set the function value for the index-th piece in this element.

Parameters:

index Position of the piece whose value is being set in the list of pieces. The first piece is numbered 1; the otherwise child (if present) is not counted, regardless of its position.

value A MathMLContentElement representing the function value to be assigned in the index-th case.

Returns :

The MathMLContentElement representing the value to be taken by the function in the index-th case.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is greater than the number of pieces in this element.

3.28.3.13 void CellML_MathMLPiecewiseElement::setOtherwise (CellML_MathMLContentElement * *otherwise*)

Set the otherwise element.

Parameters:

otherwise The MathMLContentElement representing the value to be taken by the piecewise function when none of the conditions described in the piece children is true.

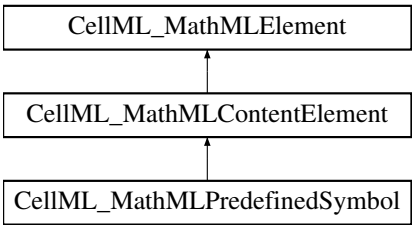
The documentation for this class was generated from the following file:

- CellML_MathMLPiecewiseElement.hpp

3.29 CellML_MathMLPredefinedSymbol Class Reference

This interface supports all of the empty built-in operator, relation, function, and constant and symbol elements that have the definitionURL and encoding attributes in addition to the standard set of attributes.

Inheritance diagram for CellML_MathMLPredefinedSymbol::



Public Methods

- **CellML_MathMLPredefinedSymbol** (const **CellML_MathMLMathElement** *ownerMathElement, const char *symbol)

  Constructor.

- ~**CellML_MathMLPredefinedSymbol** ()

  Destructor.

- const char * **getArity** () const

  Get the arity.

- const char * **getSymbolName** () const

  Get the symbol name.

Private Methods

- **CellML_MathMLPredefinedSymbol** ()

  Constructor.

- **CellML_MathMLPredefinedSymbol** (const CellML_MathMLPredefinedSymbol *other)

  Copy constructor.

3.29.1 Detailed Description

This interface supports all of the empty built-in operator, relation, function, and constant and symbol elements that have the definitionURL and encoding attributes in addition to the standard set of attributes.

The elements supported in order of their appearance in Section 4.4 [The Content Markup Elements] are: inverse, compose, ident, domain, codomain, image, quotient, exp, factorial, divide, max, min, minus, plus, power,

rem, times, root, gcd, and, or, xor, not, implies, forall, exists, abs, conjugate, arg, real, imaginary, lcm, floor, ceiling, eq, neq, gt, lt, geq, leq, equivalent, approx, factorof, ln, log, int, diff, partialdiff, divergence, grad, curl, laplacian, union, intersect, in, notin, subset, prsubset, notsubset, notprsubset, setdiff, card, cartesianproduct, sum, product, limit, tendsto, sin, cos, tan, sec, csc, cot, sinh, cosh, tanh, sech, csch, coth, arcsin, arccos, arctan, arcsec, arccsc, arccot, arcsinh, arccosh, arctanh, arcsech, arccsch, arccoth, mean, sdev, variance, median, mode, moment, determinant, transpose, selector, vectorproduct, scalarproduct, outerproduct, integers, reals, rationals, naturalnumbers, complexes, primes, exponentiale, imaginaryi, notanumber, true, false, emptyset, pi, eulergamma, and infinity.

3.29.2 Constructor & Destructor Documentation

3.29.2.1 CellML_MathMLPredefinedSymbol::CellML_MathMLPredefinedSymbol (const CellML_MathMLMathElement * *ownerMathElement*, const char * *symbol*)

Constructor.

3.29.2.2 CellML_MathMLPredefinedSymbol::~CellML_MathMLPredefinedSymbol ()

Destructor.

3.29.2.3 CellML_MathMLPredefinedSymbol::CellML_MathMLPredefinedSymbol () [private]

Constructor.

3.29.2.4 CellML_MathMLPredefinedSymbol::CellML_MathMLPredefinedSymbol (const CellML_MathMLPredefinedSymbol * *other*) [private]

Copy constructor.

3.29.3 Member Function Documentation

3.29.3.1 const char* CellML_MathMLPredefinedSymbol::getArity () const

Get the arity.

Returns :
A string representing the number of arguments. Values include 0, 1, ... and variable.

3.29.3.2 const char* CellML_MathMLPredefinedSymbol::getSymbolName () const

Get the symbol name.

This is a convenience attribute only; accessing it should be synonymous with accessing the Element::tagName attribute.

Returns :
A string giving the name of the MathML element represented.

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

- CellML_MathMLPredefinedSymbol.hpp

3.30 CellML_Model Class Reference

The CellML_Model is the top level CellML object.

Public Methods

Constructors and assignment operators.

- **CellML_Model** ()
  Default constructor for CellML_Model.
- **CellML_Model** (const CellML_Model &other)
  Copy constructor.
- CellML_Model & **operator=** (const CellML_Model &other)
  Assignment operator.

Destructor.

- ~**CellML_Model** ()
  Destructor for CellML_Model.

Equality.

- bool **equals** (const CellML_Model &other) const
  Check for equality of models.

Get functions.

- const char * **getName** () const
  Gets the name of the model.
- const **CellML_UnitsList** & **getUnitsList** () const
  Get the units list from this model.
- **CellML_UnitsList** & **getUnitsList** ()
  Get the units list from this model.
- const **CellML_ComponentList** & **getComponentList** () const
  Get the component list from this model.
- **CellML_ComponentList** & **getComponentList** ()
  Get the component list from this model.

- const **CellML_ConnectionList** & **getConnectionList** () const
  Get the connection list from this model.
- **CellML_ConnectionList** & **getConnectionList** ()
  Get the connection list from this model.
- const **CellML_GroupList** & **getGroupList** () const
  Get the group list from this model.
- **CellML_GroupList** & **getGroupList** ()
  Get the group list from this model.

Cloning function.

- CellML_Model **clone** (const bool deep) const
  Returns a duplicate of this model.

Query functions.

- bool **isValid** () const
  Test whether this model is valid.

Set functions.

- void **setName** (const char *name)
  Sets the name of the model.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML Model into a DOM node.

Public Attributes

Factory methods.

- **CellML_Units createUnits** () const
  Create a units.
- **CellML_Component createComponent** () const
  Create a component.
- **CellML_Connection createConnection** () const
  Create a connection.
- **CellML_Group createGroup** () const
  Create a group.

Static Public Attributes

- const int **numberOfStandardUnits** = 34

  The number of standard units in the model.

Private Attributes

- ModelContent * **fContent**

  The content of the model.

3.30.1 Detailed Description

The CellML_Model is the top level CellML object.

It will hold an entire model description once the model is read from a file or created programatically. A model's units list is always initialised with the dictionary of standard units defined in Section 5.2.1 of the CellML 1.0 Specification. These standard units will be marked internally to the implementation to ensure they are never removed from the model or overwritten.

3.30.2 Constructor & Destructor Documentation

3.30.2.1 CellML_Model::CellML_Model ()

Default constructor for CellML_Model.

3.30.2.2 CellML_Model::CellML_Model (const CellML_Model & *other*)

Copy constructor.

Will only make a shallow copy of the model. i.e., all the children objects will not be copied. **clone**() (p. 120) should be used if a deep copy is required.

Parameters:
other The model to be copied.

See also:
clone() (p. 120).

3.30.2.3 CellML_Model::~CellML_Model ()

Destructor for CellML_Model.

3.30.3 Member Function Documentation

3.30.3.1 CellML_Model CellML_Model::clone (const bool *deep*) const

Returns a duplicate of this model.

This function serves as a generic copy constructor for models. Cloning a model results in a new content object being created and initialised with the contents of the model being cloned.

Parameters:
deep If true, recursively clone the child objects of the model (i.e. components, connections, variables, reactions,...); if false, clone only the model itself.

Returns :
The duplicate model.

3.30.3.2 bool CellML_Model::equals (const CellML_Model & *other*) const

Check for equality of models.

This will recursively descend through this model and the other model testing for equality.

Parameters:
other The model reference with which this object is compared.

Returns :
True if both models are identical; return false otherwise.

3.30.3.3 void CellML_Model::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a model from the given DOM source node which should be a model element in the CellML namespace.

Parameters:
srcNode The DOM node containing a CellML model description.

Exceptions:
CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a model object from a DOM source.
DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.30.3.4 CellML_ComponentList& CellML_Model::getComponentList ()

Get the component list from this model.

Returns :
A reference to this model's component list.

3.30.3.5 const CellML_ComponentList& CellML_Model::getComponentList () const

Get the component list from this model.

Returns :
A constant reference to this model's component list.

3.30.3.6 CellML_ConnectionList& CellML_Model::getConnectionList ()

Get the connection list from this model.

Returns :
    A reference to this model's connection list.

3.30.3.7 const CellML_ConnectionList& CellML_Model::getConnectionList () const

Get the connection list from this model.

Returns :
    A constant reference to this model's connection list.

3.30.3.8 CellML_GroupList& CellML_Model::getGroupList ()

Get the group list from this model.

Returns :
    A reference to this model's group list.

3.30.3.9 const CellML_GroupList& CellML_Model::getGroupList () const

Get the group list from this model.

Returns :
    A constant reference to this model's group list.

3.30.3.10 const char* CellML_Model::getName () const

Gets the name of the model.

Returns a null-terminated character array.

Returns :
    The name of the model.

3.30.3.11 CellML_UnitsList& CellML_Model::getUnitsList ()

Get the units list from this model.

A model's units list is always initialised with the default units defined in section 5.2.1 of the CellML 1.0 specification.

Returns :
    A reference to this model's units list.

3.30.3.12 const CellML_UnitsList& CellML_Model::getUnitsList () const

Get the units list from this model.

A model's units list is always initialised with the default units defined in section 5.2.1 of the CellML 1.0 specification.

Returns :
A constant reference to this model's units list.

3.30.3.13 bool CellML_Model::isValid () const

Test whether this model is valid.

What does this mean for a CellML Model ?? This should only be called once the model has been fully populated and the calling routine wants to check the validity of the current contents of the model.

Returns :
True if the model is valid; false otherwise.

3.30.3.14 CellML_Model& CellML_Model::operator= (const CellML_Model & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the clone method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 120).

3.30.3.15 void CellML_Model::setName (const char * *name*)

Sets the name of the model.

Takes a copy of the name so it can be safely freed by the calling routine.

Parameters:
name The name to be assigned to this model.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the name is an illegal CellML identifier.

3.30.3.16 DOMNode* CellML_Model::toNode () const

Translate this CellML Model into a DOM node.

If the model was initially created from a DOM node, the node from which it was built will be used to initialise a new DOM node, which is then updated to the current state of the model. Otherwise an empty new element will be created to contain the current state of the model. The model is checked for validity before any translation to DOM objects occurs.

Note: The returned node will have no parent node and will belong to a dummy DOM document used to create the node returned. Depending on the DOM implementation used this document may require special destruction. For example, using the Apache Xerces-C implementation you would do something like this:

```
..
..
..
try {
  DOMNode* node = model.toNode();
} catch .... {
}
// use the node...
node->getOwnerDocument()->release();
..
..
..
```

Returns :
The DOM Node created which represents the current contents of this model object.

Exceptions:
CellML_Exception (p. 41) INVALID_CELLML_ERR: Raised when the model to be translated contains invalid CellML.

DOMException Raised if any DOM errors occur while building the node.

3.30.4 Member Data Documentation

3.30.4.1 CellML_Component CellML_Model::createComponent() const

Create a component.

Creates an empty **CellML_Component** (p. 4) object as a child of this model.

Returns :
The component object created.

3.30.4.2 CellML_Connection CellML_Model::createConnection() const

Create a connection.

Creates an empty **CellML_Connection** (p. 29) object as a child of this model.

Returns :
The connection object created.

3.30.4.3 CellML_Group CellML_Model::createGroup() const

Create a group.

Creates an empty **CellML_Group** (p. 44) object as a child of this model.

Returns :
The group object created.

3.30.4.4 CellML_Units CellML_Model::createUnits() const

Create a units.

Creates an empty **CellML_Units** (p. 149) object as a child of this model.

Returns :
The units object created.

3.30.4.5 ModelContent* CellML_Model::fContent [private]

The content of the model.

3.30.4.6 const int CellML_Model::numberOfStandardUnits = 34 [static]

The number of standard units in the model.

Sets the number of standard units in the CellML 1.0 dictionary of units -- defined in Section 5.2.1 of the CellML 1.0 Specification. The first **CellML_Model::numberOfStandardUnits** (p. 125) entries in a model's units list are always guaranteed to be the standard units.

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

- CellML_Model.hpp

3.31 CellML_RelationshipRef Class Reference

CellML_RelationshipRefs are used to specify the meaning implied by a given component grouping.

Public Methods

Constructors and assignment operator.

- **CellML_RelationshipRef** (const **CellML_Group** &parentGroup)
  Constructor for CellML_RelationshipRef.
- **CellML_RelationshipRef** (const CellML_RelationshipRef &other)
  Copy constructor.
- CellML_RelationshipRef & **operator=** (const CellML_RelationshipRef &other)
  Assignment operator.

Destructor.

- ~**CellML_RelationshipRef** ()
  Destructor for CellML_RelationshipRef.

Equality.

- bool **equals** (const CellML_RelationshipRef &other) const
  Check for equality of relationship_refs.

Get functions.

- const **CellML_Group** & **getParentGroup** () const
  Gets the parent group of this relationship_ref object.
- const char * **getRelationship** () const
  Get the relationship string.
- const char * **getRelationshipURI** () const
  Get the URI of the namespace used to specify the relationship string.
- const char * **getRelationshipName** () const
  Get the name of the relationship_ref.

Cloning function.

- CellML_RelationshipRef **clone** (const bool deep) const
  Returns a duplicate of this relationship_ref object.

Query functions.

- bool **isValid** () const
  Test whether this relationship_ref is valid.
- bool **isRelationshipSet** () const
  Test if the relationship has been set.
- bool **isRelationshipContainment** () const
  Test if the relationship is the CellML containment relationship.
- bool **isRelationshipEncapsulation** () const
  Test if the relationship is the CellML encapsulation relationship.
- bool **isRelationshipExternal** () const
  Test if the relationship is an externally defined relationship.

Set functions.

- void **setRelationshipContainment** ()
  Sets the relationship attributes for a CellML containment relationship.
- void **setRelationshipEncapsulation** ()
  Sets the relationship attributes for a CellML encapsulation relationship.
- void **setRelationship** (const char *relationship)
  Sets the relationship string.
- void **setRelationshipURI** (const char *namespaceURI)
  Sets the namepsace URI of the relationship.
- void **setRelationshipName** (const char *name)
  Sets the name of this relationship.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Constructor from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML RelationshipRef into a DOM node.

Private Methods

- **CellML_RelationshipRef** ()
  Default constructor for CellML_RelationshipRef.
- void **setParentGroup** (const **CellML_Group** &group)
  Sets the parent group of this relationship_ref object.

Private Attributes

- const **CellML_Group** * **fParentGroup**
  The parent group.
- RelationshipRefContent * **fContent**
  The content of this relationship_ref.

Friends

- class **CellML_RelationshipRefList**

3.31.1 Detailed Description

CellML_RelationshipRefs are used to specify the meaning implied by a given component grouping.

3.31.2 Constructor & Destructor Documentation

3.31.2.1 CellML_RelationshipRef::CellML_RelationshipRef (const CellML_Group & *parentGroup*)

Constructor for CellML_RelationshipRef.

3.31.2.2 CellML_RelationshipRef::CellML_RelationshipRef (const CellML_RelationshipRef & *other*)

Copy constructor.

The copy constructor will return a new CellML_RelationshipRef that has the same content object as the object being copied. If you want to have a new content object you need to use the clone method.

Parameters:
other The object to be copied.

See also:
clone() (p. 128).

3.31.2.3 CellML_RelationshipRef::~CellML_RelationshipRef ()

Destructor for CellML_RelationshipRef.

3.31.2.4 CellML_RelationshipRef::CellML_RelationshipRef () [private]

Default constructor for CellML_RelationshipRef.

3.31.3 Member Function Documentation

3.31.3.1 CellML_RelationshipRef CellML_RelationshipRef::clone (const bool *deep*) const

Returns a duplicate of this relationship_ref object.

This function serves as a generic copy constructor for relationship_refs.

Parameters:
deep If true, recursively clone the child objects of the relationship_ref; if false, clone only the relationship_ref itself.

Returns :
The duplicate relationship_ref.

3.31.3.2 bool CellML_RelationshipRef::equals (const CellML_RelationshipRef & *other*) const

Check for equality of relationship_refs.

This will test for equality between this relationship_ref and other.

Parameters:
other The relationship_ref reference with which this object is compared.

Returns :
True if both relationship_refs are identical; return false otherwise.

3.31.3.3 void CellML_RelationshipRef::fromNode (const DOMNode * *srcNode*)

Constructor from a DOM source.

Method which creates a relationship_ref from the given DOM source node which should be a relationship_ref element in the CellML namespace.

Parameters:
srcNode The DOM node containing a CellML relationship_ref description.

Exceptions:
CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a relationship_ref object from a DOM source.
DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.31.3.4 const CellML_Group& CellML_RelationshipRef::getParentGroup () const

Gets the parent group of this relationship_ref object.

Returns :
A constant reference to the parent CellML Group of this relationship_ref object.

3.31.3.5 const char* CellML_RelationshipRef::getRelationship () const

Get the relationship string.

The relationship string describes the type of the relationship. While this is most useful for applications which specify their own relationship types, the correct values will be returned for the CellML relationships of containment and encapsulation.

Returns :
A null-terminated character array representing the relationship string.

3.31.3.6 const char* CellML_RelationshipRef::getRelationshipName () const

Get the name of the relationship_ref.

Returns :
A null-terminated character array representing the name of the relationship_ref, or null if no name is defined.

3.31.3.7 const char* CellML_RelationshipRef::getRelationshipURI () const

Get the URI of the namespace used to specify the relationship string.

Returns the URI of the namespace in which the relationship string for this relationship_ref was specified. While this is most useful for applications which specify their own relationship types, the correct values will be returned for the CellML relationships of containment and encapsulation.

Returns :
A null-terminated character array representing the relationship namespace URI.

3.31.3.8 bool CellML_RelationshipRef::isRelationshipContainment () const

Test if the relationship is the CellML containment relationship.

Returns :
True if the relationship is the CellML containment relationship.

3.31.3.9 bool CellML_RelationshipRef::isRelationshipEncapsulation () const

Test if the relationship is the CellML encapsulation relationship.

Returns :
True if the relationship is the CellML encapsulation relationship.

3.31.3.10 bool CellML_RelationshipRef::isRelationshipExternal () const

Test if the relationship is an externally defined relationship.

Returns :
True if the relationship is defined externally (i.e., outside the CellML namespace); false otherwise.

3.31.3.11 bool CellML_RelationshipRef::isRelationshipSet () const

Test if the relationship has been set.

Returns :
True if the relationship has been set; false otherwise.

3.31.3.12 bool CellML_RelationshipRef::isValid () const

Test whether this relationship_ref is valid.

What does this mean for a CellML RelationshipRef ?? This should only be called once the relationship_ref has been fully populated and the calling routine wants to check the validity of the current contents of the relationship_ref.

Returns :
True if the realtionship_ref is valid; false otherwise.

3.31.3.13 CellML_RelationshipRef& CellML_RelationshipRef::operator= (const CellML_RelationshipRef & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the clone method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 128).

3.31.3.14 void CellML_RelationshipRef::setParentGroup (const CellML_Group & *group*) [private]

Sets the parent group of this relationship_ref object.

3.31.3.15 void CellML_RelationshipRef::setRelationship (const char * *relationship*)

Sets the relationship string.

Takes a copy of the relationship so it can be safely freed by the calling routine.

Parameters:
relationship The type of the relationship.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if relationship is a null string.

3.31.3.16 void CellML_RelationshipRef::setRelationshipContainment ()

Sets the relationship attributes for a CellML containment relationship.

Convenience method for setting this relationship_ref to be a CellML containment relationship.

3.31.3.17 void CellML_RelationshipRef::setRelationshipEncapsulation ()

Sets the relationship attributes for a CellML encapsulation relationship.

Convenience method for setting this relationship_ref to be a CellML encapsulation relationship.

3.31.3.18 void CellML_RelationshipRef::setRelationshipName (const char * *name*)

Sets the name of this relationship.

Takes a copy of the name so it can be safely freed by the calling routine.

Parameters:
name The name of the relationship.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the name is an illegal CellML identifier.

CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if namespaceURI is a null string.

3.31.3.19 void CellML_RelationshipRef::setRelationshipURI (const char * *namespaceURI*)

Sets the namepsace URI of the relationship.

Takes a copy of the namespaceURI so it can be safely freed by the calling routine.

Parameters:
namespaceURI The namespace URI of the relationship.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if namespaceURI is a null string.

3.31.3.20 DOMNode* CellML_RelationshipRef::toNode () const

Translate this CellML RelationshipRef into a DOM node.

Returns :
The DOM Node created which represents the current contents of this relationship_ref object.

3.31.4 Member Data Documentation

3.31.4.1 RelationshipRefContent* CellML_RelationshipRef::fContent [private]

The content of this relationship_ref.

3.31.4.2 const CellML_Group* CellML_RelationshipRef::fParentGroup [private]

The parent group.

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

- CellML_RelationshipRef.hpp

3.32 CellML_RelationshipRefList Class Reference

A class for representing lists of **CellML_RelationshipRef** (p. 125)s.

Public Methods

Constructors and assignment operators.

- **CellML_RelationshipRefList** (const **CellML_Group** &parentGroup)
  Constructor for lists belonging to groups.
- **CellML_RelationshipRefList** (const CellML_RelationshipRefList &other)
  Copy constructor.
- CellML_RelationshipRefList & **operator=** (const CellML_RelationshipRefList &other)
  Assignment operator.

Destructor.

- **~CellML_RelationshipRefList** ()
  Destructor for CellML_RelationshipRefList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_RelationshipRefList &other) const
  Test two lists are equal.
- const **CellML_RelationshipRef** & **get** (const int index) const
  Get a relationship_ref from the list.
- void **append** (**CellML_RelationshipRef** &newRelationshipRef)
  Append a relationship_ref to the list.
- void **append** (const CellML_RelationshipRefList &other)
  Append a list of relationship_ref's to this list.
- **CellML_RelationshipRef remove** (const **CellML_RelationshipRef** &oldRelationshipRef)
  Remove the given relationship_ref from the list.
- **CellML_RelationshipRef replace** (**CellML_RelationshipRef** &newRelationshipRef, const **CellML_RelationshipRef** &oldRelationshipRef)
  Replace a relationship_ref object with another.
- const **CellML_Group** & **getParentGroup** () const
  Gets the parent group of this list object.

Private Methods

- **CellML_RelationshipRefList** ()
  Default constructor.

3.32.1 Detailed Description

A class for representing lists of **CellML_RelationshipRef** (p. 125)s.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 CellML_RelationshipRefList::CellML_RelationshipRefList (const CellML_Group & *parentGroup*)

Constructor for lists belonging to groups.

Parameters:
parentGroup The parent of this list.

3.32.2.2 CellML_RelationshipRefList::CellML_RelationshipRefList (const CellML_RelationshipRefList & *other*)

Copy constructor.

Copies only references to objects in the list, not the actual objects. Note that this does not change the parent of the list or the elements of the list. If you wish to create a copy of a list to initialise a new object you should first create an empty list and then append the existing list to the new list.

Parameters:
other The list to be copied.

See also:
append(const CellML_RelationshipRefList&) (p. 134)

3.32.2.3 CellML_RelationshipRefList::~CellML_RelationshipRefList ()

Destructor for CellML_RelationshipRefList.

3.32.2.4 CellML_RelationshipRefList::CellML_RelationshipRefList () [private]

Default constructor.

Will construct an empty list.

3.32.3 Member Function Documentation

3.32.3.1 void CellML_RelationshipRefList::append (const CellML_RelationshipRefList & *other*)

Append a list of relationship_ref's to this list.

Appends the contents of other to the end of this list, ensuring that the relationship_ref's added to the list have to same parent as this list.

Parameters:
other The list of relationship_ref's to add.

3.32.3.2 void CellML_RelationshipRefList::append (CellML_RelationshipRef & *newRelationshipRef*)

Append a relationship_ref to the list.

Adds newRelationshipRef to the end of the list. Appending the relationship_ref to the list will ensure that newRelationshipRef will have the same parent as this list.

Parameters:
newRelationshipRef The relationship_ref object to add to the end of this list.

3.32.3.3 bool CellML_RelationshipRefList::equals (const CellML_RelationshipRefList & *other*) const

Test two lists are equal.

Test the contents of this and other lists for equality. The list are deemed equal if they are both the same length and for each entry in this list there is an equal entry in the other list.

Parameters:
other The list to compare this list to.

Returns :
true if this list is equal to other; false otherwise.

3.32.3.4 const CellML_RelationshipRef& CellML_RelationshipRefList::get (const int *index*) const

Get a relationship_ref from the list.

Used to get a relationship_ref object from this list by its position in the list.

Parameters:
index The index of the desired relationship_ref (valid range is from 0 to **length()** (p. 136)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the valid range for this relationship_ref list.

See also:
length() (p. 136)

3.32.3.5 const CellML_Group& CellML_RelationshipRefList::getParentGroup () const

Gets the parent group of this list object.

Returns :
A constant reference to the parent CellML Group of this list object.

3.32.3.6 bool CellML_RelationshipRefList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.32.3.7 int CellML_RelationshipRefList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.32.3.8 CellML_RelationshipRefList& CellML_RelationshipRefList::operator= (const CellML_RelationshipRefList & *other*)

Assignment operator.

Parameters:
other The source to be assigned.

3.32.3.9 CellML_RelationshipRef CellML_RelationshipRefList::remove (const CellML_RelationshipRef & *oldRelationshipRef*)

Remove the given relationship_ref from the list.

Removes oldRelationshipRef from the list and returns it.

Parameters:
oldRelationshipRef The relationship_ref object to remove from the list.

Returns :
The relationship_ref removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldRelationshipRef is not found in this list.

3.32.3.10 CellML_RelationshipRef CellML_RelationshipRefList::replace (CellML_RelationshipRef & *newRelationshipRef*, const CellML_RelationshipRef & *oldRelationshipRef*)

Replace a relationship_ref object with another.

Replaces oldRelationshipRef with newRelationshipRef. newRelationshipRef will be put into the same position in the list that is vacated by oldRelationshipRef - other than that this method is essentially the same as doing a remove(oldRelationshipRef) followed by a append(newRelationshipRef). This method will also ensure that newRelationshipRef has the same parent as this list.

Parameters:

newRelationshipRef The relationship_ref to add to the list.

oldRelationshipRef The relationship_ref to remove from the list.

Returns :

The relationship_ref removed from the list.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldRelationshipRef is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

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

- CellML_RelationshipRefList.hpp

3.33 CellML_Unit Class Reference

CellML_Unit represents a CellML description of specific use of a physical quantity unit.

Public Methods

Constructors and assignment operators.

- **CellML_Unit** (const **CellML_Units** &parentUnits)
  Units constructor.
- **CellML_Unit** (const CellML_Unit &other)
  Copy constructor.
- CellML_Unit & **operator=** (const CellML_Unit &other)
  Assignment operator.

Destructor.

- **~CellML_Unit** ()
  Destructor for **CellML_Units** (p. 149).

Equality.

- bool **equals** (const CellML_Unit &other) const
  Check for equality of Unit's.

Get functions.

- const **CellML_Units** & **getParentUnits** () const
  Gets the parent units of this unit object.
- const char * **getUnits** () const
  Gets the name of the units.
- int **getPrefix** () const
  Gets the unit's prefix.
- float **getExponent** () const
  Gets the unit's exponent.
- float **getMultiplier** () const
  Gets the unit's multiplier.
- float **getOffset** () const
  Gets the unit's offset.

Cloning function.

- CellML_Unit **clone** (const bool deep) const
  Returns a duplicate of this unit object.

Query functions.

- bool **isValid** () const
  Test whether this unit is valid.

Set functions.

- void **setUnits** (const char *units)
  Sets the name of the units used.
- void **setPrefix** (const int value)
  Set the prefix value.
- void **setPrefix** (const char *value)
  Set the prefix value.
- void **setExponent** (const float value)
  Set the exponent value.
- void **setExponent** (const char *value)
  Set the exponent value.

- void **setMultiplier** (const float value)
  Set the multiplier value.
- void **setMultiplier** (const char *value)
  Set the multiplier value.
- void **setOffset** (const float value)
  Set the offset value.
- void **setOffset** (const char *value)
  Set the offset value.

Translation functions ??.

- void **fromNode** (const DOMNode *srcNode)
  Construct from a DOM source.
- DOMNode * **toNode** () const
  Translate (??) this CellML Unit into a DOM node.

Private Methods

- **CellML_Unit** ()
  Default constructor for CellML_Unit.
- void **setParentUnits** (const **CellML_Units** &units)
  Sets the parent units of this unit object.

Private Attributes

- const **CellML_Units** * **fParentUnits**
  The parent of this unit.
- UnitContent * **fContent**
  The content of this unit.

Friends

- class **CellML_UnitList**

3.33.1 Detailed Description

CellML_Unit represents a CellML description of specific use of a physical quantity unit.

3.33.2 Constructor & Destructor Documentation

3.33.2.1 CellML_Unit::CellML_Unit (const CellML_Units & *parentUnits*)

Units constructor.

Parameters:
parentUnits The parent object of this unit.

3.33.2.2 CellML_Unit::CellML_Unit (const CellML_Unit & *other*)

Copy constructor.

The copy constructor will return a new CellML_Unit object that has the same content object as the object being copied. If you want to have a new content object you need to use the clone method.

Parameters:
other The object to be copied.

See also:
clone() (p. 140).

3.33.2.3 CellML_Unit::~CellML_Unit ()

Destructor for **CellML_Units** (p. 149).

3.33.2.4 CellML_Unit::CellML_Unit () [private]

Default constructor for CellML_Unit.

3.33.3 Member Function Documentation

3.33.3.1 CellML_Unit CellML_Unit::clone (const bool *deep*) const

Returns a duplicate of this unit object.

This function serves as a generic copy constructor for unit. In this case, the clone will be the same for both deep being true and false. Cloning a unit results in a new content object being created and initialised with the contents of the unit being cloned.

Parameters:
deep If true, recursively clone the child objects of the unit; if false, clone only the unit itself.

Returns :
The duplicate unit.

3.33.3.2 bool CellML_Unit::equals (const CellML_Unit & *other*) const

Check for equality of Unit's.

This will test for equality between this unit and other. Where two unit's are equal when all their CellML members are equal.

Parameters:
other The unit reference with which this object is compared.

Returns :
True if both units are identical; return false otherwise.

3.33.3.3 void CellML_Unit::fromNode (const DOMNode * *srcNode*)

Construct from a DOM source.

Method which creates a unit from the given DOM source node which should be a unit element.

Parameters:
srcNode The DOM node containing a CellML unit description.

Exceptions:
all

3.33.3.4 float CellML_Unit::getExponent () const

Gets the unit's exponent.

Returns :
The unit's exponent.

3.33.3.5 float CellML_Unit::getMultiplier () const

Gets the unit's multiplier.

Returns :
The unit's multiplier.

3.33.3.6 float CellML_Unit::getOffset () const

Gets the unit's offset.

Returns :
The unit's offset.

3.33.3.7 const CellML_Units& CellML_Unit::getParentUnits () const

Gets the parent units of this unit object.

Returns :
A constant reference to the parent CellML Units of this unit object.

See also:
setParentUnits() (p. 143).

3.33.3.8 int CellML_Unit::getPrefix () const

Gets the unit's prefix.

Returns :
The unit's prefix.

3.33.3.9 const char* CellML_Unit::getUnits () const

Gets the name of the units.

Returns :
A null-terminated character array containing the name of the units.

3.33.3.10 bool CellML_Unit::isValid () const

Test whether this unit is valid.

What does this mean for a CellML Unit ?? This should only be called once the unit has been fully populated and the calling routine wants to check the validity of the current contents of the unit.

Returns :
True if the unit is valid; false otherwise.

3.33.3.11 CellML_Unit& CellML_Unit::operator= (const CellML_Unit & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the clone method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 140).

3.33.3.12 void CellML_Unit::setExponent (const char * *value*)

Set the exponent value.

Checks the value for a valid exponent value and sets the prefix of this unit to that value. Valid values are real numbers.

Parameters:
value The string specifying the exponent value.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if the string value does not contain a valid exponent value.

3.33.3.13 void CellML_Unit::setExponent (const float *value*)

Set the exponent value.

Parameters:
value The value to give the exponent.

3.33.3.14 void CellML_Unit::setMultiplier (const char * *value*)

Set the multiplier value.

Checks the value for a valid multiplier value and sets the prefix of this unit to that value. Valid values are real numbers.

Parameters:
value The string specifying the multiplier value.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if the string value does not contain a valid multiplier value.

3.33.3.15 void CellML_Unit::setMultiplier (const float *value*)

Set the multiplier value.

Parameters:
value The value to give the multiplier.

3.33.3.16 void CellML_Unit::setOffset (const char * *value*)

Set the offset value.

Checks the value for a valid offset value and sets the prefix of this unit to that value. Valid values are real numbers.

Parameters:
value The string specifying the offset value.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if the string value does not contain a valid offset value.

3.33.3.17 void CellML_Unit::setOffset (const float *value*)

Set the offset value.

Parameters:
value The value to give the offset.

3.33.3.18 void CellML_Unit::setParentUnits (const CellML_Units & *units*) [private]

Sets the parent units of this unit object.

3.33.3.19 void CellML_Unit::setPrefix (const char * *value*)

Set the prefix value.

Checks the value for a valid prefix value and sets the prefix of this unit to that value. Valid values are integers and strings from the set of predefined values (milli, kilo, mega, etc...).

Parameters:
value The string specifying the prefix value.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if the string value does not contain a valid prefix value.

3.33.3.20 void CellML_Unit::setPrefix (const int *value*)

Set the prefix value.

Parameters:
value The value to give the prefix.

3.33.3.21 void CellML_Unit::setUnits (const char * *units*)

Sets the name of the units used.

Takes a copy of the units so it can be safely freed by the calling routine.

Parameters:
units The units to be assigned to this unit.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the units is an illegal CellML identifier.

3.33.3.22 DOMNode* CellML_Unit::toNode () const

Translate (??) this CellML Unit into a DOM node.

Returns :
The DOM Node created which represents the current contents of this unit object.

3.33.4 Member Data Documentation

3.33.4.1 UnitContent* CellML_Unit::fContent [private]

The content of this unit.

3.33.4.2 const CellML_Units* CellML_Unit::fParentUnits [private]

The parent of this unit.

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

- CellML_Unit.hpp

3.34 CellML_UnitList Class Reference

A class for representing lists of **CellML_Unit** (p. 137).

Public Methods

Constructors and assignment operators.

- **CellML_UnitList** (const **CellML_Units** &parentUnits)
  Constructor for lists belonging to units.
- **CellML_UnitList** (const CellML_UnitList &other)
  Copy constructor.
- CellML_UnitList & **operator=** (const CellML_UnitList &other)
  Assignment operator.

Destructor.

- ~**CellML_UnitList** ()
  Destructor for CellML_UnitList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_UnitList &other) const
  Test two lists are equal.
- const **CellML_Unit** & **get** (const int index) const
  Get a unit from the list.
- void **append** (**CellML_Unit** &newUnit)
  Append a unit to the list.
- void **append** (const CellML_UnitList &other)
  Append a list of unit's to this list.
- **CellML_Unit remove** (const **CellML_Unit** &oldUnit)
  Remove the given unit from the list.
- **CellML_Unit replace** (**CellML_Unit** &newUnit, const **CellML_Unit** &oldUnit)
  Replace a unit object with another.
- const **CellML_Units** & **getParentUnits** () const
  Gets the parent units of this list object.

Private Methods

- **CellML_UnitList** ()
  Default constructor.

3.34.1 Detailed Description

A class for representing lists of **CellML_Unit** (p. 137).

3.34.2 Constructor & Destructor Documentation

3.34.2.1 CellML_UnitList::CellML_UnitList (const CellML_Units & *parentUnits*)

Constructor for lists belonging to units.

Parameters:
parentUnits The parent of this list.

3.34.2.2 CellML_UnitList::CellML_UnitList (const CellML_UnitList & *other*)

Copy constructor.

Will create a new list and populate it with clones of the units in the source list.

Parameters:
other The list to be copied.

See also:
CellML_Unit::CellML_Unit(const CellML_Unit&) (p. 140) , **CellML_Unit::clone**() (p. 140) , **append**(const CellML_UnitList&) (p. 146)

3.34.2.3 CellML_UnitList::~CellML_UnitList ()

Destructor for CellML_UnitList.

3.34.2.4 CellML_UnitList::CellML_UnitList () [private]

Default constructor.

Will construct an empty list.

3.34.3 Member Function Documentation

3.34.3.1 void CellML_UnitList::append (const CellML_UnitList & *other*)

Append a list of unit's to this list.

Appends clones of the unit's in the source list to this list, ensuring that the unit's added to the list have the same parent as this list.

Parameters:
other The list of unit's to add.

See also:
CellML_Unit::clone() (p. 140).

3.34.3.2 void CellML_UnitList::append (CellML_Unit & *newUnit*)

Append a unit to the list.

Adds newUnit to the end of the list. Appending the units to the list will ensure that newUnit will have the same parent as this list.

Parameters:
newUnit The unit object to add to the end of this list.

3.34.3.3 bool CellML_UnitList::equals (const CellML_UnitList & *other*) const

Test two lists are equal.

Test the contents of this and other lists for equality. The list are deemed equal if they are both the same length and for each entry in this list there is an equal entry in the other list.

Parameters:
other The list to compare this list to.

Returns :
true if this list is equal to other; false otherwise.

3.34.3.4 const CellML_Unit& CellML_UnitList::get (const int *index*) const

Get a unit from the list.

Used to get a units object from this list by its position in the list.

Parameters:
index The index of the desired units (valid range is from 0 to **length()** (p. 148)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the valid range for this units list.

See also:
length() (p. 148)

3.34.3.5 const CellML_Units& CellML_UnitList::getParentUnits () const

Gets the parent units of this list object.

Returns :
A constant reference to the parent CellML Units of this list object.

3.34.3.6 bool CellML_UnitList::isEmpty () const

Test if the list contains any objects.

Returns :
    true if the list is empty; false otherwise.

3.34.3.7 int CellML_UnitList::length () const

Get the size of the list.

Returns :
    The number of objects in the list.

3.34.3.8 CellML_UnitList& CellML_UnitList::operator= (const CellML_UnitList & *other*)

Assignment operator.

Will assign a copy of the source list to the destination list.

Parameters:
    other The source to be assigned.

See also:
    CellML_UnitList(const CellML_UnitList&) (p. 146)

3.34.3.9 CellML_Unit CellML_UnitList::remove (const CellML_Unit & *oldUnit*)

Remove the given unit from the list.

Removes oldUnit from the list and returns it.

Parameters:
    oldUnit The unit object to remove from the list.

Returns :
    The unit removed from the list.

Exceptions:
    CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldUnit is not found in this list.

3.34.3.10 CellML_Unit CellML_UnitList::replace (CellML_Unit & *newUnit*, const CellML_Unit & *oldUnit*)

Replace a unit object with another.

Replaces oldUnit with newUnit. newUnit will be put into the same position in the list that is vacated by oldUnit - other than that this method is essentially the same as doing a remove(oldUnit) followed by a append(newUnit). This method will also ensure that newUnit has the same parent as this list.

Parameters:

newUnit The unit to add to the list.

oldUnit The unit to remove from the list.

Returns :

The unit removed from the list.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldUnit is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

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

- CellML_UnitList.hpp

3.35 CellML_Units Class Reference

CellML_Units represents a CellML description of physical units.

Public Methods

Constructors and assignment operators.

- **CellML_Units** (const **CellML_Model** &parentModel)
  Constructor for a units belonging to a model.
- **CellML_Units** (const **CellML_Component** &parentComponent)
  Constructor for a units belonging to a component.
- **CellML_Units** (const CellML_Units &other)
  Copy constructor.
- CellML_Units & **operator=** (const CellML_Units &other)
  Assignment operator.

Destructor.

- **~CellML_Units** ()
  Destructor for CellML_Units.

Equality.

- bool **equals** (const CellML_Units &other) const
  Check for equality of units.

Get functions.

- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this units object.

- const **CellML_Component** & **getParentComponent** () const
  Gets the parent component of this units object.
- const char * **getName** () const
  Gets the name of the units.
- const bool **getBaseUnits** () const
  Get the base units flag.
- const **CellML_UnitList** & **getUnitList** () const
  Get the unit list from this units.
- **CellML_UnitList** & **getUnitList** ()
  Get the unit list from this units.

Cloning function.

- CellML_Units **clone** (const bool deep=false) const
  Returns a duplicate of this units object.

Query functions.

- bool **isValid** () const
  Test whether this units is valid.
- bool **hasParentComponent** () const
  Determine if this units has a component parent.
- bool **hasParentModel** () const
  Determine if this units has a model parent.

Set functions.

- void **setName** (const char *name)
  Sets the name of the units.
- void **setBaseUnits** (const bool value)
  Set the base units flag.
- void **setBaseUnits** (const char *value)
  Set the base units flag.

Translation function ??.

- void **fromNode** (const DOMNode *srcNode)
  Construct from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML Units into a DOM node.

Public Attributes

- `bool` **standardUnits**

  Need some way to distinguish standard units from others.

Factory methods.

- **CellML_Unit createUnit** `()` `const`

  Create a unit.

Private Methods

- **CellML_Units** `()`

  Default constructor for CellML_Units.

- `void` **setParentModel** `(const` **CellML_Model** `&model)`

  Sets the parent model of this units object.

- `void` **setParentComponent** `(const` **CellML_Component** `&component)`

  Sets the parent component of this units object.

Private Attributes

- `UnitsContent *` **fContent**

  The content of the units.

Friends

- `class` **CellML_UnitsList**

3.35.1 Detailed Description

CellML_Units represents a CellML description of physical units.

Complex units can be described by adding multiple CellML Unit children to a CellML_Units object.

3.35.2 Constructor & Destructor Documentation

3.35.2.1 CellML_Units::CellML_Units (const CellML_Model & *parentModel*)

Constructor for a units belonging to a model.

Parameters:
parentModel The parent of this units.

3.35.2.2 CellML_Units::CellML_Units (const CellML_Component & *parentComponent*)

Constructor for a units belonging to a component.

Parameters:
parentComponent The parent of this units.

3.35.2.3 CellML_Units::CellML_Units (const CellML_Units & *other*)

Copy constructor.

The copy constructor will return a new CellML_Units object that has the same content object as the object being copied. If you want to have a new content object you need to use the clone method.

Parameters:
other The object to be copied.

See also:
clone() (p. 152).

3.35.2.4 CellML_Units::~CellML_Units ()

Destructor for CellML_Units.

3.35.2.5 CellML_Units::CellML_Units () [private]

Default constructor for CellML_Units.

3.35.3 Member Function Documentation

3.35.3.1 CellML_Units CellML_Units::clone (const bool *deep* = false) const

Returns a duplicate of this units object.

This function serves as a generic copy constructor for units.

Parameters:
deep If true, recursively clone the child objects of the units; if false, clone only the units itself.

Returns :
The duplicate units.

3.35.3.2 bool CellML_Units::equals (const CellML_Units & *other*) const

Check for equality of units.

This will recursively descend through this units and the other units testing for equality.

Parameters:
other The units reference with which this object is compared.

Returns :
True if both units are identical; return false otherwise.

3.35.3.3 void CellML_Units::fromNode (const DOMNode * *srcNode*)

Construct from a DOM source.

Constructor method which creates a units from the given DOM source node which should be a units element.

Parameters:
srcNode The DOM node containing a CellML units description.

Exceptions:
all

3.35.3.4 const bool CellML_Units::getBaseUnits () const

Get the base units flag.

Returns :
The state of the base units flag.

3.35.3.5 const char* CellML_Units::getName () const

Gets the name of the units.

Returns :
The name of the units.

3.35.3.6 const CellML_Component& CellML_Units::getParentComponent () const

Gets the parent component of this units object.

Returns :
A constant reference to the parent CellML Component of this units object.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised when this units object does not have a component parent.

See also:
getParentModel() (p. 154),**setParentModel()** (p. 156),**setParentComponent()** (p. 156).

3.35.3.7 const CellML_Model& CellML_Units::getParentModel () const

Gets the parent model of this units object.

Returns :
A constant reference to the parent CellML Model of this units object.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised when this units object does not have a model parent.

See also:
getParentComponent() (p. 153), **setParentModel()** (p. 156), **setParentComponent()** (p. 156).

3.35.3.8 CellML_UnitList& CellML_Units::getUnitList ()

Get the unit list from this units.

Returns :
A reference to this units' unit list.

3.35.3.9 const CellML_UnitList& CellML_Units::getUnitList () const

Get the unit list from this units.

Returns :
A constant reference to this units' unit list.

3.35.3.10 bool CellML_Units::hasParentComponent () const

Determine if this units has a component parent.

Returns :
true if this units has a **CellML_Component** (p. 4) parent; false otherwise.

See also:
hasParentModel() (p. 154).

3.35.3.11 bool CellML_Units::hasParentModel () const

Determine if this units has a model parent.

Returns :
true if this units has a **CellML_Model** (p. 118) parent; false otherwise.

See also:
hasParentComponent() (p. 154).

3.35.3.12 bool CellML_Units::isValid () const

Test whether this units is valid.

What does this mean for a CellML Units ?? This should only be called once the units has been fully populated and the calling routine wants to check the validity of the current contents of the units.

Returns :
True if the units is valid; false otherwise.

3.35.3.13 CellML_Units& CellML_Units::operator= (const CellML_Units & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the clone method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 152).

3.35.3.14 void CellML_Units::setBaseUnits (const char * *value*)

Set the base units flag.

Looks for the string "yes" or "no" in value and sets the base units flag to true if "yes" is found, or false for "no".

Parameters:
value The string specifying the base units flag.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if the string value is neither "yes" nor "no" (case sensitive).

3.35.3.15 void CellML_Units::setBaseUnits (const bool *value*)

Set the base units flag.

Parameters:
value The value to set the base units flag to.

3.35.3.16 void CellML_Units::setName (const char * *name*)

Sets the name of the units.

Takes a copy of the name so it can be safely freed by the calling routine.

Parameters:
name The name to be assigned to this model.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the name is an illegal CellML identifier.

3.35.3.17 void CellML_Units::setParentComponent (const CellML_Component & *component*) [private]

Sets the parent component of this units object.

Setting the parent component will also ensure that the parent model is set to null as a units can either have a component parent or a model parent but not both. **CellML_UnitsList** (p. 157) objects should be the only objects calling this method.

Parameters:
component The parent component of this units.

See also:
setParentComponent() (p. 156),**getParentModel()** (p. 154),**getParentComponent()** (p. 153).

3.35.3.18 void CellML_Units::setParentModel (const CellML_Model & *model*) [private]

Sets the parent model of this units object.

Setting the parent model will also ensure that the parent component is set to null as a units can either have a model parent or a component parent but not both. **CellML_UnitsList** (p. 157) objects should be the only objects calling this method.

Parameters:
model The parent model of this units.

See also:
setParentComponent() (p. 156),**getParentModel()** (p. 154),**getParentComponent()** (p. 153).

3.35.3.19 DOMNode* CellML_Units::toNode () const

Translate this CellML Units into a DOM node.

If the units was initially created from a DOM node, the node from which it was built will be used to initialise a new DOM node, which is then updated to the current state of the units. Otherwise an empty new element will be created to contain the current state of the units. The units is checked for validity before any translation to DOM objects occurs.

Note: The returned node will have no parent node and will belong to a dummy DOM document used to create the node returned. Depending on the DOM implementation used this document may require special destruction. For example, using the Apache Xerces-C implementation you would do something like this:

```
..
..
..
try {
  DOMNode* node = model.toNode();
} catch .... {
}
// use the node...
node->getOwnerDocument()->release();
..
..
..
```

Returns :
The DOM Node created which represents the current contents of this units object.

Exceptions:
CellML_Exception (p. 41) INVALID_CELLML_ERR: Raised when the units to be translated contains invalid CellML.
DOMException Raised if any DOM errors occur while building the node.

3.35.4 Member Data Documentation

3.35.4.1 CellML_Unit CellML_Units::createUnit() const

Create a unit.

Creates an empty **CellML_Unit** (p. 137) object as a child of this units.

Returns :
The unit object created.

3.35.4.2 UnitsContent* CellML_Units::fContent [private]

The content of the units.

3.35.4.3 bool CellML_Units::standardUnits

Need some way to distinguish standard units from others.

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

- CellML_Units.hpp

3.36 CellML_UnitsList Class Reference

A class for representing lists of **CellML_Units** (p. 149).

Public Methods

Constructors and assignment operators.

- **CellML_UnitsList** (const **CellML_Model** &parentModel)
  Constructor for lists belonging to models.
- **CellML_UnitsList** (const **CellML_Component** &parentComponent)
  Constructor for lists belonging to components.
- **CellML_UnitsList** (const CellML_UnitsList &other)
  Copy constructor.
- CellML_UnitsList & **operator=** (const CellML_UnitsList &other)
  Assignment operator.

Destructor.

- **~CellML_UnitsList** ()
  Destructor for CellML_UnitsList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_UnitsList &other) const
  Test two lists are equal.
- const **CellML_Units** & **get** (const int index) const
  Get a units from the list.
- const **CellML_Units** & **get** (const char *name) const
  Get a units from the list.
- void **append** (**CellML_Units** &newUnits)
  Append a units to the list.
- void **append** (const CellML_UnitsList &other)
  Append a list of units to this list.
- **CellML_Units remove** (const **CellML_Units** &oldUnits)
  Remove the given units from the list.
- **CellML_Units replace** (**CellML_Units** &newUnits, const **CellML_Units** &oldUnits)
  Replace a units object with another.
- const **CellML_Model** & **getParentModel** () const
  Gets the parent model of this list object.
- const **CellML_Component** & **getParentComponent** () const
  Gets the parent component of this list object.

Private Methods

- **CellML_UnitsList** ()
  Default constructor.

3.36.1 Detailed Description

A class for representing lists of **CellML_Units** (p. 149).

Enforces the CellML requirement for unique names within the list.

3.36.2 Constructor & Destructor Documentation

3.36.2.1 CellML_UnitsList::CellML_UnitsList (const CellML_Model & *parentModel*)

Constructor for lists belonging to models.

Parameters:
parentModel The parent of this list.

3.36.2.2 CellML_UnitsList::CellML_UnitsList (const CellML_Component & *parentComponent*)

Constructor for lists belonging to components.

Parameters:
parentComponent The parent of this list.

3.36.2.3 CellML_UnitsList::CellML_UnitsList (const CellML_UnitsList & *other*)

Copy constructor.

Will create a new list and populate it with clones of the units in the source list.

Parameters:
other The list to be copied.

See also:
CellML_Units::CellML_Units(const CellML_Units&) (p. 152) , **CellML_Units::clone**() (p. 152) , **append**(const CellML_UnitsList&) (p. 160)

3.36.2.4 CellML_UnitsList::~CellML_UnitsList ()

Destructor for CellML_UnitsList.

3.36.2.5 CellML_UnitsList::CellML_UnitsList () [private]

Default constructor.

Will construct an empty list.

3.36.3 Member Function Documentation

3.36.3.1 void CellML_UnitsList::append (const CellML_UnitsList & *other*)

Append a list of units to this list.

Appends clones of the units in the source list to this list, ensuring that the units added to the list have the same parent as this list.

Parameters:
other The list of units to add.

Exceptions:
CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if any units in the other list have the same name as a units already in the list.

See also:
CellML_Units::clone() (p. 152).

3.36.3.2 void CellML_UnitsList::append (CellML_Units & *newUnits*)

Append a units to the list.

Adds newUnits to the end of the list. Appending the units to the list will ensure that newUnits will have the same parent as this list.

Parameters:
newUnits The units object to add to the end of this list.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if newUnits has an invalid name.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if a units with the same name as newUnits is already in the list.

CellML_Exception (p. 41) STANDARD_UNITS_ERR: Raised if newUnits has the name of a standard units.

3.36.3.3 bool CellML_UnitsList::equals (const CellML_UnitsList & *other*) const

Test two lists are equal.

Test the contents of this and other lists for equality. The list are deemed equal if they are both the same length and for each entry in this list there is an equal entry in the other list.

Parameters:
other The list to compare this list to.

Returns :
true if this list is equal to other; false otherwise.

3.36.3.4 const CellML_Units& CellML_UnitsList::get (const char * *name*) const

Get a units from the list.

Used to get a units object from this list by its name. If a units with the given name is not found in this units list and this units list is a child of a component, then the parent component's parent model units list is then queried for a matching units.

Parameters:
name The name of the desired units.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if name is not a name of one of the units currently in this units list (or the parent model's units list).

3.36.3.5 const CellML_Units& CellML_UnitsList::get (const int *index*) const

Get a units from the list.

Used to get a units object from this list by its position in the list.

Parameters:
index The index of the desired units (valid range is from 0 to **length()** (p. 162)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the valid range for this units list.

See also:
length() (p. 162)

3.36.3.6 const CellML_Component& CellML_UnitsList::getParentComponent () const

Gets the parent component of this list object.

Returns :
A constant reference to the parent CellML Component of this list object.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised when this list object does not have a component parent.

See also:
getParentModel() (p. 162).

3.36.3.7 const CellML_Model& CellML_UnitsList::getParentModel () const

Gets the parent model of this list object.

Returns :
A constant reference to the parent CellML Model of this list object.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised when this list object does not have a model parent.

See also:
getParentComponent() (p. 161).

3.36.3.8 bool CellML_UnitsList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.36.3.9 int CellML_UnitsList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.36.3.10 CellML_UnitsList& CellML_UnitsList::operator= (const CellML_UnitsList & *other*)

Assignment operator.

Will assign a copy of the source list to the destination list.

Parameters:
other The source to be assigned.

See also:
CellML_UnitsList(const CellML_UnitsList&) (p. 159)

3.36.3.11 CellML_Units CellML_UnitsList::remove (const CellML_Units & *oldUnits*)

Remove the given units from the list.

Removes oldUnits from the list and returns it.

Parameters:
oldUnits The units object to remove from the list.

Returns :
The units removed from the list.

Exceptions:

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldUnits is not found in this list.

CellML_Exception (p. 41) STANDARD_UNITS_ERR: Raised if you are trying to remove a standard units from a model's units list.

3.36.3.12 CellML_Units CellML_UnitsList::replace (CellML_Units & *newUnits*, const CellML_Units & *oldUnits*)

Replace a units object with another.

Replaces oldUnits with newUnits. newUnits will be put into the same position in the list that is vacated by oldUnits - other than that this method is essentially the same as doing a remove(oldUnits) followed by a append(newUnits). This method will also ensure that newUnits has the same parent as this list.

Parameters:

newUnits The units to add to the list.

oldUnits The units to remove from the list.

Returns :

The units removed from the list.

Exceptions:

CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if newUnits has an invalid name. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if a units with the same name as newUnits is already in the list (after oldUnits is removed from the list). If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldUnits is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) STANDARD_UNITS_ERR: Raised if you are trying to remove a standard units from a model's units list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) STANDARD_UNITS_ERR: Raised if newUnits has the name of a standard units.

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

- CellML_UnitsList.hpp

3.37 CellML_Variable Class Reference

A CellML_Variable is the most elemental of the CellML classes.

Public Methods

Constructors and assignment operators.

- **CellML_Variable** (const **CellML_Component** &parentComponent)
  Component constructor for CellML_Variable.
- **CellML_Variable** (const CellML_Variable &other)
  Copy constructor.
- CellML_Variable & **operator=** (const CellML_Variable &other)
  Assignment operator.

Destructor.

- **~CellML_Variable** ()
  Destructor for CellML_Variable.

Equality.

- bool **equals** (const CellML_Variable &other) const
  Check for equality of variables.

Get functions.

- const **CellML_Component** & **getParentComponent** () const
  Gets the parent component of this variable object.
- const char * **getName** () const
  Gets the name of the variable.
- const char * **getUnitsName** () const
  Gets the name of the units of the variable.
- float **getInitialValue** () const
  Get the variable's initial value.
- const CellML_Variable & **getConnectedVariable** () const
  Get the variable this variable is connected to.

Cloning function.

- CellML_Variable **clone** (const bool deep) const
  Returns a duplicate of this variable object.

Query functions.

- bool **isValid** () const
  Test whether this variable is valid.

- bool **isInitialValueSet** () const
  Test whether the initial value has been set for this variable.
- bool **isPublicInterfaceIn** () const
  Test whether the public interface for this variable is "in".
- bool **isPublicInterfaceOut** () const
  Test whether the public interface for this variable is "out".
- bool **isPublicInterfaceNone** () const
  Test whether the public interface for this variable is "none".
- bool **isPrivateInterfaceIn** () const
  Test whether the private interface for this variable is "in".
- bool **isPrivateInterfaceOut** () const
  Test whether the private interface for this variable is "out".
- bool **isPrivateInterfaceNone** () const
  Test whether the private interface for this variable is "none".

Set functions.

- void **setName** (const char *name)
  Sets the name of the variable.
- void **setUnitsName** (const char *unitsName)
  Sets the name of the units for this variable.
- void **setPublicInterfaceIn** ()
  Set the public interface to "in".
- void **setPublicInterfaceOut** ()
  Set the public interface to "out".
- void **setPublicInterfaceNone** ()
  Set the public interface to "none".
- void **setPublicInterface** (const char *value)
  Sets the public interface of this variable.
- void **setPrivateInterfaceIn** ()
  Set the private interface to "in".
- void **setPrivateInterfaceOut** ()
  Set the private interface to "out".
- void **setPrivateInterfaceNone** ()
  Set the private interface to "none".
- void **setPrivateInterface** (const char *value)
  Sets the private interface of this variable.

- void **setInitialValue** (float value)
  Sets the initial value of this variable.
- void **setInitialValue** (const char *value)
  Sets the initial value of this variable.

Translation functions.

- void **fromNode** (const DOMNode *srcNode)
  Construct from a DOM source.
- DOMNode * **toNode** () const
  Translate this CellML Variable into a DOM node.

Other functions.

- const CellML_Variable & **resolve** () const
  Resolve this variable.

Private Methods

- **CellML_Variable** ()
  Default constructor for CellML_Variable.
- void **setParentComponent** (const **CellML_Component** &component)
  Sets the parent component of this variable object.

Private Attributes

- const **CellML_Component** * **fParentComponent**
  The parent of this variable.
- VariableContent * **fContent**
  The content of the variable.

Friends

- class **CellML_VariableList**

3.37.1 Detailed Description

A CellML_Variable is the most elemental of the CellML classes.

Models are basically definitions of the interactions between variables.

Maybe all the connected variable stuff should be handled in a more transparent manner. For example, the calling application would say "set this variable to

this value" and the implementation would then be responsible for making sure that the value of the connected variable is then set...The trouble is that by using constant references everywhere means the implementation would have to do all the work of getting a copy of the connected variable, modifying it, and putting it back - still preferable to expecting the user to manually track through to the variable they should really be changing ??

3.37.2 Constructor & Destructor Documentation

3.37.2.1 CellML_Variable::CellML_Variable (const CellML_Component & *parentComponent*)

Component constructor for CellML_Variable.

3.37.2.2 CellML_Variable::CellML_Variable (const CellML_Variable & *other*)

Copy constructor.

The copy constructor will return a new CellML_Variable object that has the same content object as the object being copied. If you want to have a new content object you need to use the clone method.

Parameters:
other The object to be copied.

See also:
clone() (p. 167).

3.37.2.3 CellML_Variable::~CellML_Variable ()

Destructor for CellML_Variable.

3.37.2.4 CellML_Variable::CellML_Variable () [private]

Default constructor for CellML_Variable.

3.37.3 Member Function Documentation

3.37.3.1 CellML_Variable CellML_Variable::clone (const bool *deep*) const

Returns a duplicate of this variable object.

This function serves as a generic copy constructor for variables. In this case, the clone will be the same for both deep being true and false. Cloning a variable results in a new content object being created and initialised with the contents of the variable being cloned.

Parameters:
deep If true, recursively clone the child objects of the variable; if false, clone only the variable itself.

Returns :
The duplicate variable.

3.37.3.2 bool CellML_Variable::equals (const CellML_Variable & *other*) const

Check for equality of variables.

This will test for equality between this variable and other.

Parameters:
other The variable reference with which this object is compared.

Returns :
True if both variables are identical; return false otherwise.

3.37.3.3 void CellML_Variable::fromNode (const DOMNode * *srcNode*)

Construct from a DOM source.

Method which creates a variable from the given DOM source node which should be a variable element in the CellML namespace.

Parameters:
srcNode The DOM node containing a CellML variable description.

Exceptions:
CellML_Exception (p. 41) Pretty much the whole bunch of CellML exceptions are possible when constructing a component object from a DOM source.
DOMException Raised if any DOM errors occur while trying to save non-CellML nodes.

3.37.3.4 const CellML_Variable& CellML_Variable::getConnectedVariable () const

Get the variable this variable is connected to.

Variable's with a private or public interface value of 'in' need to be *"connected"* to a output variable to have any real meaning. This function returns that variable (if there is one).

Returns :
A constant reference to the connected variable.

3.37.3.5 float CellML_Variable::getInitialValue () const

Get the variable's initial value.

This function will always return a value, **isInitialValueSet()** (p. 169) should be used to determine if the value has any meaning.

Returns :
The initial value of the variable.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if this function is called when **isInitialValueSet()** (p. 169) returns a value of false.

See also:
isInitialValueSet() (p. 169).

3.37.3.6 const char* CellML_Variable::getName () const

Gets the name of the variable.

Returns :
A null-terminated string representation of the name of this variable.

3.37.3.7 const CellML_Component& CellML_Variable::getParentComponent () const

Gets the parent component of this variable object.

Returns :
A constant reference to the parent CellML Component of this variable object.

3.37.3.8 const char* CellML_Variable::getUnitsName () const

Gets the name of the units of the variable.

Returns :
A null-terminated string representation of the name of the units of this variable.

3.37.3.9 bool CellML_Variable::isInitialValueSet () const

Test whether the initial value has been set for this variable.

Returns :
True if the initial value has been set; false otherwise.

See also:
getInitialValue() (p. 168).

3.37.3.10 bool CellML_Variable::isPrivateInterfaceIn () const

Test whether the private interface for this variable is *"in"*.

Returns :
True if the private interface for this variable is *"in"*; false otherwise.

3.37.3.11 bool CellML_Variable::isPrivateInterfaceNone () const

Test whether the private interface for this variable is *"none"*.

Returns :
True if the private interface for this variable is *"none"*; false otherwise.

3.37.3.12 bool CellML_Variable::isPrivateInterfaceOut () const

Test whether the private interface for this variable is *"out"*.

Returns :
True if the private interface for this variable is *"out"*; false otherwise.

3.37.3.13 bool CellML_Variable::isPublicInterfaceIn () const

Test whether the public interface for this variable is *"in"*.

Returns :
True if the public interface for this variable is *"in"*; false otherwise.

3.37.3.14 bool CellML_Variable::isPublicInterfaceNone () const

Test whether the public interface for this variable is *"none"*.

Returns :
True if the public interface for this variable is *"none"*; false otherwise.

3.37.3.15 bool CellML_Variable::isPublicInterfaceOut () const

Test whether the public interface for this variable is *"out"*.

Returns :
True if the public interface for this variable is *"out"*; false otherwise.

3.37.3.16 bool CellML_Variable::isValid () const

Test whether this variable is valid.

What does this mean for a CellML Variable ?? This should only be called once the variable has been fully populated and the calling routine wants to check the validity of the current contents of the variable.

Returns :
True if the variable is valid; false otherwise.

3.37.3.17 CellML_Variable& CellML_Variable::operator= (const CellML_Variable & *other*)

Assignment operator.

The object being assigned to will have its content set to that of the object being assigned. To have a new content object created you need to use the clone method.

Parameters:
other The source to be assigned.

See also:
clone() (p. 167).

3.37.3.18 const CellML_Variable& CellML_Variable::resolve () const

Resolve this variable.

If this variable has a public or private interface of "in", the variable this one is connected to will be returned. If this variable does not have a public or private interface of "in", this variable is returned.

Returns :
The resolved variable.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if this variable can not be resolved using the information currently found in the model which owns it.

3.37.3.19 void CellML_Variable::setInitialValue (const char * *value*)

Sets the initial value of this variable.

Checks the value string for a real number and uses that value to set the initial value of this variable.

Parameters:
value The string representation of the initial value given to this variable.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if value does not provide a single real number.

3.37.3.20 void CellML_Variable::setInitialValue (float *value*)

Sets the initial value of this variable.

Parameters:
value The initial value to give this variable.

3.37.3.21 void CellML_Variable::setName (const char * *name*)

Sets the name of the variable.

Takes a copy of the name so it can be safely freed by the calling routine.

Parameters:
name The name to be assigned to this variable.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the name is an illegal CellML identifier.

3.37.3.22 void CellML_Variable::setParentComponent (const CellML_Component & *component*) [private]

Sets the parent component of this variable object.

3.37.3.23 void CellML_Variable::setPrivateInterface (const char * *value*)

Sets the private interface of this variable.

Checks the value string for a value of *"in"*, *"out"*, or *"none"* and sets the private interface accordingly.

Parameters:
value The string representation of the value given to the private interface of this variable.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if value is not *"in"*, *"out"*, or *"none"*.

3.37.3.24 void CellML_Variable::setPrivateInterfaceIn ()

Set the private interface to *"in"*.

3.37.3.25 void CellML_Variable::setPrivateInterfaceNone ()

Set the private interface to *"none"*.

3.37.3.26 void CellML_Variable::setPrivateInterfaceOut ()

Set the private interface to *"out"*.

3.37.3.27 void CellML_Variable::setPublicInterface (const char * *value*)

Sets the public interface of this variable.

Checks the value string for a value of *"in"*, *"out"*, or *"none"* and sets the public interface accordingly.

Parameters:
value The string representation of the value given to the public interface of this variable.

Exceptions:
CellML_Exception (p. 41) INVALID_VALUE_ERR: Raised if value is not *"in"*, *"out"*, or *"none"*.

3.37.3.28 void CellML_Variable::setPublicInterfaceIn ()

Set the public interface to *"in"*.

3.37.3.29 void CellML_Variable::setPublicInterfaceNone ()

Set the public interface to *"none"*.

3.37.3.30 void CellML_Variable::setPublicInterfaceOut ()

Set the public interface to *"out"*.

3.37.3.31 void CellML_Variable::setUnitsName (const char * *unitsName*)

Sets the name of the units for this variable.

Takes a copy of the units so it can be safely freed by the calling routine.

Parameters:
unitsName The name of the units to be assigned to this variable.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if the unitsName is an illegal CellML identifier.

3.37.3.32 DOMNode* CellML_Variable::toNode () const

Translate this CellML Variable into a DOM node.

Returns :
The DOM Node created which represents the current contents of this variable object.

3.37.4 Member Data Documentation

3.37.4.1 VariableContent* CellML_Variable::fContent [private]

The content of the variable.

3.37.4.2 const CellML_Component* CellML_Variable::fParentComponent [private]

The parent of this variable.

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

- CellML_Variable.hpp

3.38 CellML_VariableList Class Reference

A class for representing lists of **CellML_Variable** (p. 163)'s.

Public Methods

Constructors and assignment operators.

- **CellML_VariableList** (const **CellML_Component** &parentComponent)
  Constructor for lists belonging to components.
- **CellML_VariableList** (const CellML_VariableList &other)
  Copy constructor.
- CellML_VariableList & **operator=** (const CellML_VariableList &other)
  Assignment operator.

Destructor.

- **~CellML_VariableList** ()
  Destructor for CellML_VariableList.

Methods.

- int **length** () const
  Get the size of the list.
- bool **isEmpty** () const
  Test if the list contains any objects.
- bool **equals** (const CellML_VariableList &other) const
  Test two lists are equal.
- const **CellML_Variable** & **get** (const int index) const
  Get a variable from the list.
- const **CellML_Variable** & **get** (const char *name) const
  Get a variable from the list.
- void **append** (**CellML_Variable** &newVariable)
  Append a variable to the list.
- void **append** (const CellML_VariableList &other)
  Append a list of variables to this list.
- **CellML_Variable remove** (const **CellML_Variable** &oldVariable)
  Remove the given variable from the list.
- **CellML_Variable replace** (**CellML_Variable** &newVariable, const **CellML_Variable** &oldVariable)
  Replace a variable object with another.
- const **CellML_Component** & **getParentComponent** () const
  Gets the parent component of this list object.

Private Methods

- **CellML_VariableList** ()
  Default constructor.

3.38.1 Detailed Description

A class for representing lists of **CellML_Variable** (p. 163)'s.

Enforces the CellML requirement for unique names within the list.

3.38.2 Constructor & Destructor Documentation

3.38.2.1 CellML_VariableList::CellML_VariableList (const CellML_Component & *parentComponent*)

Constructor for lists belonging to components.

Parameters:
parentComponent The parent of this list.

3.38.2.2 CellML_VariableList::CellML_VariableList (const CellML_VariableList & *other*)

Copy constructor.

Copies only references to objects in the list, not the actual objects. Note that this does not change the parent of the list or the elements of the list. If you wish to create a copy of a list to initialise a new object you should first create an empty list and then append the existing list to the new list.

Parameters:
other The list to be copied.

See also:
append(const CellML_VariableList&) (p. 175)

3.38.2.3 CellML_VariableList::~CellML_VariableList ()

Destructor for CellML_VariableList.

3.38.2.4 CellML_VariableList::CellML_VariableList () [private]

Default constructor.

Will construct an empty list.

3.38.3 Member Function Documentation

3.38.3.1 void CellML_VariableList::append (const CellML_VariableList & *other*)

Append a list of variables to this list.

Appends the contents of other to the end of this list, ensuring that the variable added to the list have the same parent as this list.

Parameters:
other The list of variables to add.

Exceptions:
CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if any variable in the other list have the same name as a variable already in the list.

3.38.3.2 void CellML_VariableList::append (CellML_Variable & *newVariable*)

Append a variable to the list.

Adds newVariable to the end of the list. Appending the variable to the list will ensure that newVariable will have the same parent as this list.

Parameters:
newVariable The variable object to add to the end of this list.

Exceptions:
CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if newVariable has an invalid name.
CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if a variable with the same name as newVariable is already in the list.

3.38.3.3 bool CellML_VariableList::equals (const CellML_VariableList & *other*) const

Test two lists are equal.

Test the contents of this and other lists for equality. The lists are deemed equal if they are both the same length and for each entry in this list there is an equal entry in the other list.

Parameters:
other The list to compare this list to.

Returns :
true if this list is equal to other; false otherwise.

3.38.3.4 const CellML_Variable& CellML_VariableList::get (const char * *name*) const

Get a variable from the list.

Used to get a variable object from this list by its name.

Parameters:
name The name of the desired variable.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if name is not a name of a variable currently in this variable list.

3.38.3.5 const CellML_Variable& CellML_VariableList::get (const int *index*) const

Get a variable from the list.

Used to get a variable object from this list by its position in the list.

Parameters:
index The index of the desired variable (valid range is from 0 to **length()** (p. 177)-1.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if index is outside the valid range for this variable list.

See also:
length() (p. 177)

3.38.3.6 const CellML_Component& CellML_VariableList::getParentComponent () const

Gets the parent component of this list object.

Returns :
A constant reference to the parent CellML Component of this list object.

3.38.3.7 bool CellML_VariableList::isEmpty () const

Test if the list contains any objects.

Returns :
true if the list is empty; false otherwise.

3.38.3.8 int CellML_VariableList::length () const

Get the size of the list.

Returns :
The number of objects in the list.

3.38.3.9 CellML_VariableList& CellML_VariableList::operator= (const CellML_VariableList & *other*)

Assignment operator.

Parameters:
other The source to be assigned.

3.38.3.10 CellML_Variable CellML_VariableList::remove (const CellML_Variable & *oldVariable*)

Remove the given variable from the list.

Removes oldVariable from the list and returns it.

Parameters:
oldVariable The variable object to remove from the list.

Returns :
The variable removed from the list.

Exceptions:
CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldVariable is not found in this list.

3.38.3.11 CellML_Variable CellML_VariableList::replace (CellML_Variable & *newVariable*, const CellML_Variable & *oldVariable*)

Replace a variable object with another.

Replaces oldVariable with newVariable. newVariable will be put into the same position in the list that is vacated by oldVariable - other than that this method is essentially the same as doing a remove(oldVariable) followed by a append(newVariable). This method will also ensure that newVariable has the same parent as this list.

Parameters:

newVariable The variable to add to the list.

oldVariable The variable to remove from the list.

Returns :

The variable removed from the list.

Exceptions:

CellML_Exception (p. 41) INVALID_NAME_ERR: Raised if newVariable has an invalid name. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) INVALID_MODIFICATION_ERR: Raised if a variable with the same name as newVariable is already in the list (after oldVariable is removed from the list). If this exception is thrown the list will be unchanged from its state prior to the call to this function.

CellML_Exception (p. 41) NOT_FOUND_ERR: Raised if oldVariable is not found in this list. If this exception is thrown the list will be unchanged from its state prior to the call to this function.

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

- CellML_VariableList.hpp

Index