



Contents

1 Introduction

This manual is a reference for the `libsbml` application programming interface (API). `libsbml` is

2.1 AlgebraicRule.h

AlgebraicRule_t * AlgebraicRule_create (void)

Creates a new AlgebraicRule and returns a pointer to it.

AlgebraicRule_t * AlgebraicRule_createWith (const char *formula)

Creates a new AlgebraicRule with the given formula and returns a pointer to it. This convenience function is functionally equivalent to:

```
AlgebraicRule_t ar = AlgebraicRule_create();    Rule_setFormula((Rule_t) ar, formula);
```

2.2 AssignmentRule.h

Creates a new AssignmentRule and returns a pointer to it.

In L1 AssignmentRule is an abstract class. It exists solely to provide fields to its subclasses: CompartmentVolumeRule, ParameterRule and SpeciesConcentrationRule.

In L2 the three subclasses are gone and AssignmentRule is a concrete class. It creates a new AssignmentRule and returns a pointer to it.

2.3 ASTNode.h

int ASTNode





2.5 CompartmentVolumeRule.h

2.7 Event.h

Event


```
int Event_isSetTimeUnits (const Event_t *e)
```

Returns 1 if the timeUnits of this Event has been set, 0 otherwise.

ListOf_t * Event_getListOfEventAssignments (const Event_t *e)

Returns the list of EventAssignments for this Event.

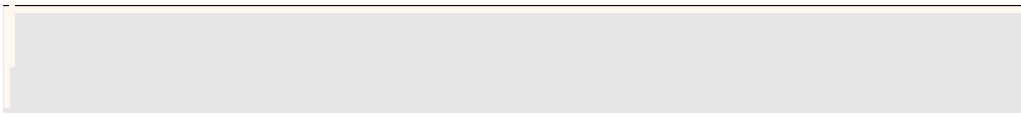
EventAssignment * Event_getEventAssignment (const Event_t *e, unsigned int n)

Returns the nth EventAssignment of this Event.

unsigned int Event_getNumEventAssignments (const Event_t *e)

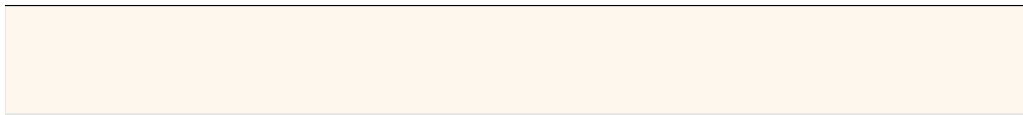
Returns the number of EventAssignments in this Event.

2.10 FormulaTokenizer.h

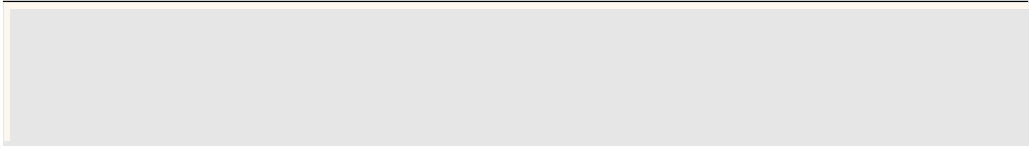




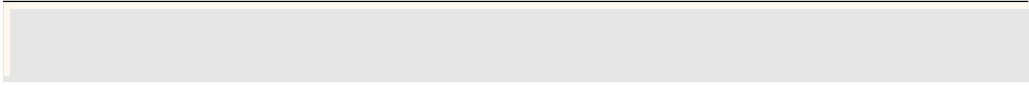
2.15 MathMLReader.h







Parameter_



2.17ModifierSpeciesReference.-31h



2.18 ParameterRule.h

2.19 Parameter.h

Parameter_t * Parameter_create (void)

Creates a new Parameter and returns a pointer to it.

Parameter_t * Parameter_createWith (const char *sid, double value, const char *units)

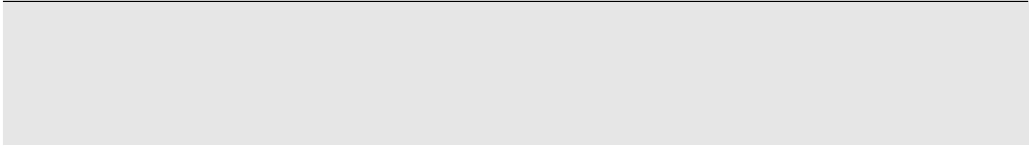
Creates a new Parameter with the given id, value and units and returns a pointer to it. This convenience function is functionally equivalent to:

```
Parameter_t p = Parameter_create();           Parameter_setId(p, id);
```

```
        _setValue(p, value); ...
```

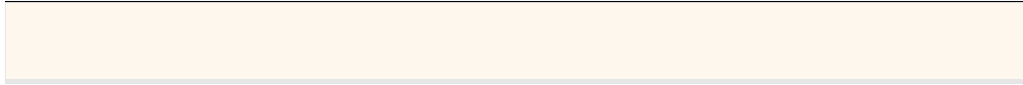
void Parameter_free (Parameter_t *p)

Frees the given Parameter.



2.21 RateRule.h

2.22 Reaction.h



int Reaction_isSetKineticLaw (const Reaction_t *r)

Returns 1 if the KineticLaw of this Reaction has been set, 0 otherwise.

int Reaction_isSetFast (const Reaction_t *r)

Returns 1 if the fast status of this Reaction has been set, 0 otherwise.

In L1, fast is optional with a default of false, which means it is effectively always set.

SpeciesReference_t * Reaction_getReactant (const Reaction_t *r, unsigned int n)

Returns the nth reactant (SpeciesReference) of this Reaction.

SpeciesReference_t * Reaction_getProduct (const Reaction_t *r, unsigned int n)

Returns the nth product (SpeciesReference) of this Reaction.

2.23 Rule.h

```
void Rule_init (Rule_t *r, SBMLTypeCode_
```

2.24 RuleType.h

RuleType_t RuleType_forName (const char *name)

Returns the RuleType with the given name (case-insensitive).

2.25 SBase.h

void SBase_init (SBase_t *sb, SBMLTypeCode_t tc)

SBase "objects" are abstract, i.e., they are not created. Rather, specific "subclasses" are created (e.g., Model) and their SBASE_FIELDS are initialized with this function. The type of the specific "subclass" is indicated by the given SBMLTypeCode.

void SBase_clear (SBase_t *sb)

Clears (frees) only the SBASE_FIELDS of sb.

SBMLTypeCode_t SBase_getTypeCode (const SBase_t *sb)

Returns the type of this SBML object.

const char * SBase_getMetaId (const SBase_t *sb)

Returns the metaid for this SBML object.

const char * SBase_getNotes (const SBase_t *sb)

Returns the notes for this SBML object.

const char * SBase_getAnnotation (const SBase_t *sb)

Returns the annotation for this SBML object.

int SBase_isSetMetaId (const SBase_t *sb)

Returns 1 if the metaid for this SBML object has been set, 0 otherwise.

int SBase_isSetNotes (const SBase_t *sb)

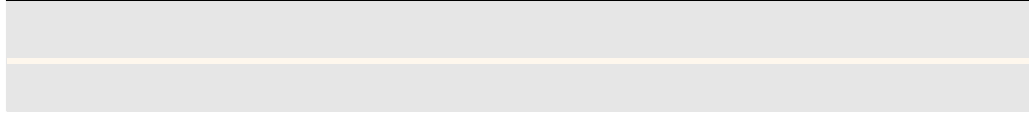
Returns 1 if the notes for this SBML object has been set, 0 otherwise.

int SBase_isSetAnnotation (const SBase_t *sb)

t *sb, SBMLTypeCode_t tc) SBase_init(SBase_t *sb, SBMLTypeCode_t tc) SBase_clear(SBase_t *sb) SBase_getTypeCode(const SBase_t *sb) SBase_getMetaId(const SBase_t *sb) SBase_getNotes(const SBase_t *sb) SBase_getAnnotation(const SBase_t *sb) SBase_isSetMetaId(const SBase_t *sb) SBase_isSetNotes(const SBase_t *sb) SBase_isSetAnnotation(const SBase_t *sb)

void SBase_unsetMetaId (SBase_t *sb)

Unsets the metaId for this SBML object. This is equivalent to:
safe_free(sb->metaId); s->metaId = NULL;



2.26 SBMLDocument.h

SBMLDocument_t * SBMLDocument_create (void)

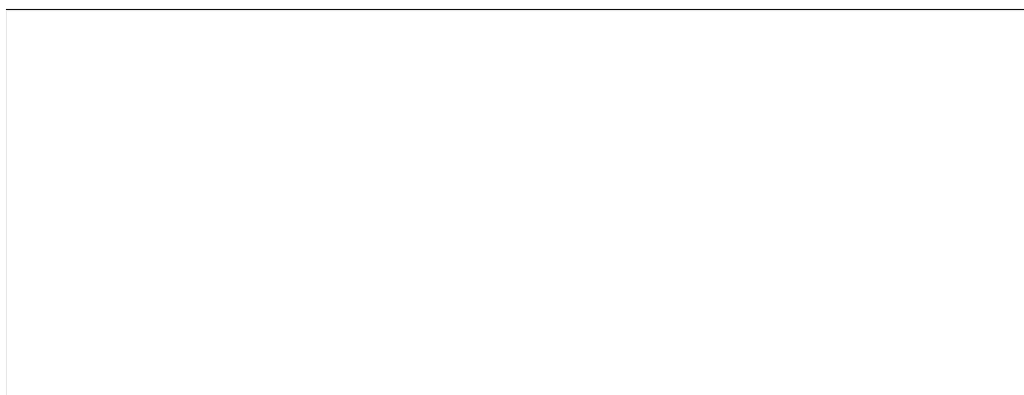
Creates a new SBMLDocument and returns a pointer to it.
The SBML level defaults to 2 and version defaults to 1.

SBMLDocument_t * SBMLDocument_createWith (unsigned int level, unsigned int version)

Creates a new SBMLDocument with the given level and version.

2.27 SBMLReader.h





2.28 SBMLWriter.h



2.31 SpeciesReference.h

```
SpeciesReference_t * SpeciesReference_create (void)
```

2.32 Species.h

Species_t * Species_create (void)

Creates a new Species and returns a pointer to it.

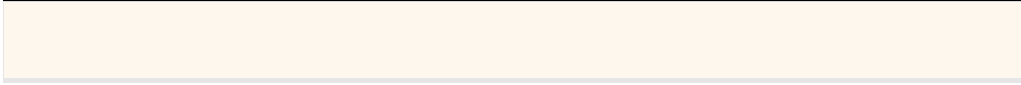
Species_t * Species_


```
void Species_setConstant (Species_t *s, int value)
```

Sets the constant field of this Species to value (boolean).

```
void Species_unsetName (Species_t *s)
```

2.33 UnitDefinition.h



void UnitDefinition

2.34 UnitKind.h

2.35 Unit.h

Unit


```
void Unit.setScale (Unit_t *u, int value)
```

Sets the scale of this Unit to the given value.

```
void Unit.setMultiplier (Unit_t *u, double value)
```

Sets the multiplier of this Unit to the given value.

```
void Unit.setOffset (Unit_t *u, double value)
```

Sets the offset of this Unit to the given value.

