libsbml API Reference Manual

Contents

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_
```

2.3 ASTNode.h

ASTNode

double ASTNode_getReal (const ASTNode_t *node)

void ASTNode_setCharacter (ASTNode_t *node, char value)

2.4 Compartment.h

Compartment_t * Compartment_create (void)

Creates a new Compartment and returns a pointer to it.

void Compartment_setUnits (Compartment_t *c, const char *sid)

Sets the units of this Compartment to a copy of sid.

void Compartment_setOutside (Compartment_t *c, const char *sid)

2.7 Event.h

Event_t * Event_create (void)

Creates a new Event and returns a pointer to it.

2.9 FormulaParser.h

2.10 FormulaTokenizer.h

FormulaTokenizer_

2.12 KineticLaw.h

KineticLaw_t * KineticLaw_

2.[redacted]List.h

2.14 ListOf.h

2.15 MathMLDocument.h

MathMLDocument_t * MathMLDocument_create (void)

Creates a new MathMLDocument and returns a pointer to it.

void MathMLDocument_free (MathMLDocument_t *d)

Frees the given MathMLDocument.

2.16 MathMLReader.h

```
CompartmentVolumeRule_t * Model_createCompartmentVolumeRule (Model_
```

Parameter_t * Model_createKineticLawParameter (Model_t *m)

Species_t * Model_getSpecies (const Model_t *m, unsigned int n)

Returns the nth Species of this Model.

Species_t * Model_getSpeciesById (const Model_

unsigned int Model_getNumSpeciesWithBoundaryCondition (const Model_t *m)

Returns the number of Species in this Model with boundaryCondition set to true.

unsigned int Model_getNumParameters (const Model_t *m)

2.18 ModifierSpeciesReference.h

2.19 ParameterRule.h

ParameterRule_t * ParameterRule_

2.20 Parameter.h

int Parameter_isSetValue (const Parameter_t *p)

Returns 1 if the value of this Parameter has been set, 0 otherwise.
In SBML L1v1, a Parameter value is required and therefore should always be set. In L1v2 and beyond, a value is optional and as such may or may not be set.

2.21 ParseMessage.h

2.23 Reaction.h

Creates a new Reaction and returns a pointer to it.

Reaction_t * Reaction_create (void)

SpeciesReference_t * Reaction_getReactantById (const Reaction_t *r, const char *sid)

int ReactionIdCmp (const char *sid, const Reaction_t *r)

The ReactionIdCmp function compares the string sid to r-¿id.
Returnss an integer less than, equal to, er greater than zero if sid is found to be, respectively, less than, to match er be greater than r-¿id. Returns -1 if e1317.66ther s1317.66d er r-¿id is NULL.

2.26 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

void SBase_setNotes (SBase_

void SBMLDocument_validate (const SBMLDocument_t *d)

Performs semantic validation on the document. Query the results by calling SBMLDocument_getNumWarnings(), SBMLDocument_

2.28 SBMLReader.h

2.31 SpeciesConcentrationRule.h

SpeciesConcentrationRule_t * SpeciesConcentrationRule_

2.32 SpeciesReference.h

2.33 Species.h

2.35 StringBuffer.h

```
StringBuffer_
```

char * StringBuffer_getBuffer (const StringBuffer_t *sb)

Returns the underlying buffer contained in this StringBuffer.

The buffer is not owned by the caller and should not be modified or deleted. The caller

2.36 UnitDefinition.h

2.37 UnitKind.h

int UnitKind_equals (UnitKind_t uk1, UnitKind_t uk2)

Tests for logical equality between two UnitKinds. This function behaves exactly like C's == operator, except for the following two cases:

- UNIT_KIND_LITER == UNIT_KIND_LITRE - UNIT_KIND_METER == UNIT_KIND_METRE

where C would yield false (since each of the above is a distinct enumeration value),

equals(.. yields true.

2.38 Unit.h

void Unit_setScale (Unit_t *u, int value)

Sets the scale of this Unit to the given value.

2.39 util.h

double util_NegZero (void)

Returns IEEE-754 Negative Zero.

int util_isInf (double d)

Returns -1 if d represents negative infinity, 1 if d represents positive infinity and 0 otherwise.