Package ‘rsbml’

April 18, 2025

Version 2.67.0

Title R support for SBML, using libsbml

Author Michael Lawrence <`michafla@gene.com`>

Depends R (>= 2.6.0), BiocGenerics (>= 0.3.2), methods, utils

Imports BiocGenerics, graph, utils

SystemRequirements libsbml (==5.10.2)

Maintainer Michael Lawrence <`michafla@gene.com`>

Description Links R to libsbml for SBML parsing, validating output, provides an S4 SBML DOM, converts SBML to R graph objects. Optionally links to the SBML ODE Solver Library (SOSLib) for simulating models.

License Artistic-2.0

URL `http://www.sbml.org`

biocViews GraphAndNetwork, Pathways, Network

git_url https://git.bioconductor.org/packages/rsbml

git_branch devel

git_last_commit 7671d4c

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-04-17

Contents

`AlgebraicRule-class` *SBML type "AlgebraicRule"*

Description

Expresses equations that are not assignments nor rates of change.

Instantiation

Objects can be created by calls of the form `new("AlgebraicRule", ...)`.

Slots

`math`**:** Object of class `"expression"` specifying the equation.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"Rule"`, directly. Class `"SBase"`, by class "Rule", distance 2.

Methods

No methods defined with class "AlgebraicRule" in the signature.

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`AssignmentRule-class` *SBML type "AssignmentRule"*

Description

An equation that assigns a value to the quantity of a `Species`, the size of a `Compartment` or the value of a `Parameter`.

Instantiation

Objects can be created by calls of the form `new("AssignmentRule", ...)`.

Slots

- `variable`**:** Object of class `"character"` naming the variable (the id of a `Species`, `Compartment` or `Parameter`) to set.
- `type`**:** Object of class `"character"`, deprecated.
- `math`**:** Object of class `"expression"` specifying the equation.
- `metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`Rule`", directly. Class "`SBase`", by class "Rule", distance 2.

Methods

- **variable** `signature(object = "AssignmentRule")`: gets the `variable` slot
- **variable<-** `signature(object = "AssignmentRule")`: sets the `variable` slot
- **type** `signature(object = "AssignmentRule")`: gets the type slot
- **type<-** `signature(object = "AssignmentRule")`: sets the type slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`BoundingBox-class` *SBML type "BoundingBox"*

Description

Species the size and position of an SBML layout object.

Instantiation

Objects can be created by calls of the form `new("BoundingBox", ...)`.

Slots

`id`**:** Object of class `"character"` uniquely identifying this component.

`position`**:** Object of class `"Point"` specifying the positon.

`dimensions`**:** Object of class `"Dimensions"` specifying the size.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "BoundingBox")`: gets the `id` slot

id<- `signature(object = "BoundingBox")`: sets the `id` slot

dimensions `signature(object = "BoundingBox")`: gets the `dimensions` slot

dimensions<- `signature(object = "BoundingBox")`: sets the `dimensions` slot

position `signature(object = "BoundingBox")`: gets the `position` slot

position<- `signature(object = "BoundingBox")`: sets the `position` slot

Author(s)

Michael Lawrence

References

`http://projects.villa-bosch.de/bcb/sbml`

`Compartment-class` *SBML type "Compartment"*

Description

A bounded space that contains `Species`.

Instantiation

Objects can be created by calls of the form `new("Compartment", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`name`: Object of class `"character"` naming this component.

`spatialDimensions`: Object of class `"integer"` indicating the number of dimensions (0, 1, 2, or 3)

`size`: Object of class `"numeric"` indicating the size in the given units.

`units`: Object of class `"character"` indicating the units (built-in or the id of a `UnitDefinition`).

`outside`: Object of class `"character"` identifying the compartment containing this compartment.

`constant`: Object of class `"logical"` indicating whether the size changes during simulation.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "Compartment")`: gets the `id` slot

id<- `signature(object = "Compartment")`: sets the `id` slot

name `signature(object = "Compartment")`: gets the `name` slot

name<- `signature(object = "Compartment")`: sets the `name` slot

constant `signature(object = "Compartment")`: gets the `constant` slot

constant<- `signature(object = "Compartment")`: sets the `constant` slot

outside `signature(object = "Compartment")`: gets the `outside` slot

outside<- `signature(object = "Compartment")`: sets the `outside` slot

size `signature(object = "Compartment")`: gets the `size` slot

size<- `signature(object = "Compartment")`: sets the `size` slot

units `signature(object = "Compartment")`: gets the `constant` slot

units<- `signature(object = "Compartment")`: sets the `constant` slot

spatialDimensions `signature(object = "Compartment")`: gets the `spatialDimensions` slot

spatialDimensions<- `signature(object = "Compartment")`: sets the `spatialDimensions` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`CompartmentGlyph-class`

SBML type "CompartmentGlyph"

Description

A glyph representing a `Compartment`.

Instantiation

Objects can be created by calls of the form `new("CompartmentGlyph", ...)`.

Slots

- `compartment`: Object of class `"character"` identifying the compartment this glyph represents.
- `id`: Object of class `"character"` uniquely identifying this component.
- `boundingBox`: Object of class `"BoundingBox"` describing the position and size of the graphical object.
- `metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`: Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"GraphicalObject"`, directly. Class `"SBase"`, by class "GraphicalObject", distance 2.

Methods

compartment `signature(object = "CompartmentGlyph")`: gets the `compartment` slot

compartment<- `signature(object = "CompartmentGlyph")`: sets the `compartment` slot

Author(s)

Michael Lawrence

References

`http://projects.villa-bosch.de/bcb/sbml`

`CompartmentType-class` *SBML Type "CompartmentType"*

Description

Declares a type of `Compartment`. Compartments with the same type are logically similar.

Objects from the Class

Objects can be created by calls of the form `new("CompartmentType", ...)`.

Slots

`id`**:** Object of class `"character"` uniquely identifying this component.

`name`**:** Object of class `"character"` naming this component.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "CompartmentType")`: gets the `id` slot

id<- `signature(object = "CompartmentType")`: sets the `id` slot

name `signature(object = "CompartmentType")`: gets the `name` slot

name<- `signature(object = "CompartmentType")`: sets the `name` slot

Note

Requires libsbml >= 3.0

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

See Also

`Compartment`

`CompartmentVolumeRule-class`

SBML type "CompartmentVolumeRule"

Description

Obsolete way to assign a volume to a `Compartment`.

Instantiation

Objects can be created by calls of the form `new("CompartmentVolumeRule", ...)`.

Slots

`compartment`**:** Object of class `"character"` identifying the compartment

`variable`**:** Object of class `"character"`, ignored.

`type`**:** Object of class `"character"`, deprecated.

`math`**:** Object of class `"expression"` specifying the equation.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"AssignmentRule"`, directly. Class `"Rule"`, by class "AssignmentRule", distance 2. Class `"SBase"`, by class "AssignmentRule", distance 3.

Methods

compartment `signature(object = "CompartmentVolumeRule")`: gets the `compartment` slot

compartment<- `signature(object = "CompartmentVolumeRule")`: sets the `compartment` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`Constraint-class` *SBML Type "Constraint"*

Description

A constraint that must be continuously satisfied throughout the simulation of a model. Once a constraint is no longer met, the simulation must halt.

Objects from the Class

Objects can be created by calls of the form `new("Constraint", ...)`.

Slots

`math`**:** Object of class `"expression"` that evaluates to `FALSE` if the constraint is not satisfied, otherwise evaluates to `TRUE`.

`message`**:** Object of class `"character"`, formatted in XHTML, that is displayed to the user by an application when the constraint is not satisfied.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

math `signature(domain = "Constraint")`: gets the `math` slot.

math<- `signature(object = "Constraint")`: sets the `math` slot.

msg `signature(domain = "Constraint")`: gets the `msg` slot.

msg<- `signature(object = "Constraint")`: sets the `msg` slot.

Note

Requires libsbml >= 3.0

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`CubicBezier-class` *SBML type "CubicBezier"*

Description

A cubic bezier curve in an SBML layout.

Instantiation

Objects can be created by calls of the form `new("CubicBezier", ...)`.

Slots

`basePoint1`**:** Object of class `"Point"` indicating the position of the base point closest to the starting point.

`basePoint2`**:** Object of class `"Point"` indicating the position of the base point farthest from the starting point.

`start`**:** Object of class `"Point"` ~~

`end`**:** Object of class `"Point"` ~~

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`LineSegment`", directly. Class "`SBase`", by class "LineSegment", distance 2.

Methods

basePoint1 `signature(object = "CubicBezier")`: gets the `basePoint1` slot

basePoint1<- `signature(object = "CubicBezier")`: sets the `basePoint1` slot

basePoint2 `signature(object = "CubicBezier")`: gets the `basePoint2` slot

basePoint2<- `signature(object = "CubicBezier")`: sets the `basePoint2` slot

Author(s)

Michael Lawrence

References

http://projects.villa-bosch.de/bcb/sbml

`Curve-class` *SBML type "Curve"*

Description

A curve (list of line segments) in an SBML layout.

Instantiation

Objects can be created by calls of the form `new("Curve", ...)`.

Slots

`curveSegments`**:** Object of class `"list"` containing the `LineSegment`s that compose the curve.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

curveSegments `signature(object = "Curve")`: gets the `curveSegments` slot

curveSegments<- `signature(object = "Curve")`: sets the `curveSegments` slot

Author(s)

Michael Lawrence

References

http://projects.villa-bosch.de/bcb/sbml

`CVTerm-class` *SBML Type "CVTerm"*

Description

A MIRIAM annotation, consisting of a qualifier ("model", "biological" or something else) and a resource (URI).

Objects from the Class

Objects can be created by calls of the form `new("CVTerm", ...)`.

Slots

- `qualifierType`**:** Object of class `"character"` specifying the type of qualifier for this term. Types "model" and "biological" have special meaning, but any string may be specified.
- `modelQualifierType`**:** Object of class `"character"` specifying the type of model qualifier, if `qualifierType` is set to "model". Types "is" and "isDescribedBy" are formally defined in MIRIAM, but any string may be specified.
- `biologicalQualifierType`**:** Object of class `"character"` specifying the type of biological qualifier, if `qualifierType` is set to "biological". Types "is", "hasPart", "isPartOf", "isVersionOf", "hasVersion", "isHomologTo", and "isDescribedBy" are formally defined in MIRIAM, though any string may be specified.
- `resources`**:** Object of class `"character"` specifying a URI that identifies some resource related an SBML element by the qualifier.

Methods

- **biologicalQualifierType** `signature(object = "CVTerm")`: gets the `biologicalQualifierType` slot.
- **biologicalQualifierType<-** `signature(object = "CVTerm")`: sets the `biologicalQualifierType` slot.
- **modelQualifierType** `signature(object = "CVTerm")`: gets the `modelQualifierType` slot.
- **modelQualifierType<-** `signature(object = "CVTerm")`: sets the `modelQualifierType` slot.
- **qualifierType** `signature(object = "CVTerm")`: gets the `qualifierType` slot.
- **qualifierType<-** `signature(object = "CVTerm")`: sets the `qualifierType` slot.
- **resources** `signature(object = "CVTerm")`: gets the `resources` slot.
- **resources<-** `signature(object = "CVTerm")`: sets the `resources` slot.

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`Delay-class` *SBML Type "Delay"*

Description

The length of time between the `Trigger`ing of an `Event` and the execution of its `EventAssignment`s.

Objects from the Class

Objects can be created by calls of the form `new("Delay", ...)`.

Slots

`math`: Object of class `"expression"` that evaluates to a quantity of time.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

math `signature(domain = "Delay")`: gets the `math` slot.

math<- `signature(object = "Delay")`: sets the `math` slot.

Note

Requires libsbml >= 3.0

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

See Also

`Event`

`describe` *Describing objects*

Description

Each class in the SBML DOM extends the `Describable` class and thus has a `describe` method, which describes an object with a short string. This is used by the `show` method to output terse textual representations of the DOM.

Usage

```
describe(object, ...)
```

Arguments

`object`	The object to be described.
`...`	Additional arguments for methods.

Value

A short textual (string) representation of `object`.

Describable objects

An object that extends `Describable` has a method for the `describe` generic, and by default `Describable` objects are `shown` by printing the output of `describe`. Note that `Describable` is a virtual tag class, no objects may be created from it.

Describable methods

show `signature(object = "Describable")`: outputs the return value of `describe`.

Author(s)

Michael Lawrence

`Dimensions-class` *SBML type "Dimensions"*

Description

Holds the size of an SBML layout object.

Instantiation

Objects can be created by calls of the form `new("Dimensions", ...)`.

Slots

- `width`: Object of class `"numeric"` indicating the width, in pixels
- `height`: Object of class `"numeric"` indicating the height, in pixels
- `depth`: Object of class `"numeric"` indicating the depth, in pixels
- `metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`: Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

- **depth** `signature(object = "Dimensions")`: gets the depth slot
- **depth<-** `signature(object = "Dimensions")`: sets the depth slot
- **height** `signature(object = "Dimensions")`: gets the `height` slot
- **height<-** `signature(object = "Dimensions")`: sets the `height` slot
- **width** `signature(object = "Dimensions")`: gets the `width` slot
- **width<-** `signature(object = "Dimensions")`: sets the `width` slot

Author(s)

Michael Lawrence

References

http://projects.villa-bosch.de/bcb/sbml

`Event-class` *SBML type "Event"*

Description

Description of a instantaneous, discontinuous change in the model state.

Instantiation

Objects can be created by calls of the form `new("Event", ...)`.

Slots

- `id`**:** Object of class `"character"` uniquely identifying this component.
- `name`**:** Object of class `"character"` naming this component.
- `trigger`**:** Object of class `"expression"` that evaluates to `TRUE` when the event is to be fired.
- `eventDelay`**:** Object of class `"expression"` that evaluates to the time until execution of this event after it has been fired.
- `timeUnits`**:** Object of class `"character"` identifying the units of the delay.
- `eventAssignments`**:** Object of class `"list"` containing `EventAssignment`s that are performed at execution.
- `useValuesFromTriggerTime`**:** Object of class `"logical"`. If `FALSE`, the event is evaluated after the delay, rather than when the event is executed.
- `metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

- **id** `signature(object = "Event")`: gets the `id` slot
- **id<-** `signature(object = "Event")`: sets the `id` slot
- **name** `signature(object = "Event")`: gets the `name` slot
- **name<-** `signature(object = "Event")`: sets the `name` slot
- **timeUnits** `signature(object = "Event")`: gets the `timeUnits` slot
- **timeUnits<-** `signature(object = "Event")`: sets the `timeUnits` slot
- **eventDelay** `signature(x = "Event")`: ...
- **eventDelay<-** `signature(object = "Event")`: sets the `delay` slot
- **eventAssignments** `signature(object = "Event")`: gets the `eventAssignments` slot
- **eventAssignments<-** `signature(object = "Event")`: sets the `eventAssignments` slot
- **trigger** `signature(object = "Event")`: gets the `trigger` slot
- **trigger<-** `signature(object = "Event")`: sets the `trigger` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`EventAssignment-class` *SBML type "EventAssignment"*

Description

As part of an event, assigns a value to the quantity of a `Species`, the size of a `Compartment` or the value of a `Parameter`.

Instantiation

Objects can be created by calls of the form `new("EventAssignment", ...)`.

Slots

`variable`**:** Object of class `"character"` ~~

`math`**:** Object of class `"expression"` that evaluates to the value to assign.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

math `signature(object = "EventAssignment")`: gets the `math` slot

math<- `signature(object = "EventAssignment")`: sets the `math` slot

variable `signature(object = "EventAssignment")`: gets the `variable` slot

variable<- `signature(object = "EventAssignment")`: sets the `variable` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`Experiment-class` *Experiment*

Description

This class is an abstraction for an experiment, e.g. in a simulation. An experiment consists of a `ExperimentProtocol`, `ExperimentDesign`, `ExperimentSubject` and `ExperimentResult`.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

- `protocol`**:** Object of empty virtual class `ExperimentProtocol`, how the experiment was or is to be performed.
- `design`**:** Object of empty virtual class `ExperimentDesign`, the design of the experiment.
- `subject`**:** Object of empty virtual class `ExperimentSubject`, the object being observed by the experiment.
- `result`**:** Object of empty virtual class `ExperimentResult`, the result of the experiment.

Methods

- **design** `signature(object = "Experiment")`: Gets the `design` slot.
- **design<-** `signature(object = "Experiment")`: Sets the `design` slot.
- **protocol** `signature(object = "Experiment")`: Gets the `protocol` slot.
- **protocol<-** `signature(object = "Experiment")`: Sets the `protocol` slot.
- **result** `signature(object = "Experiment")`: Gets the `result` slot.
- **result<-** `signature(object = "Experiment")`: Sets the `result` slot.
- **subject** `signature(object = "Experiment")`: Gets the `subject` slot.
- **subject<-** `signature(object = "Experiment")`: Sets the `subject` slot.

Author(s)

Michael Lawrence

See Also

`SOSExperiment`, an implementation that simulates SBML modules using the SBML ODE Solver library.

`FunctionDefinition-class`

SBML type "FunctionDefinition"

Description

Identifies a mathematical expression so that it may be referenced in other expressions.

Instantiation

Objects can be created by calls of the form `new("FunctionDefinition", ...)`.

Slots

`id`**:** Object of class `"character"` uniquely identifying this component.

`name`**:** Object of class `"character"` naming this component.

`math`**:** Object of class `"expression"` that defines the function.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "FunctionDefinition")`: gets the `id` slot

id<- `signature(object = "FunctionDefinition")`: sets the `id` slot

name `signature(object = "FunctionDefinition")`: gets the `name` slot

name<- `signature(object = "FunctionDefinition")`: sets the `name` slot

math `signature(object = "FunctionDefinition")`: gets the `math` slot

math<- `signature(object = "FunctionDefinition")`: sets the `math` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`GraphicalObject-class` *SBML type "GraphicalObject"*

Description

The base class for graphical objects (e.g. glyphs) in SBML layouts.

Instantiation

Objects can be created by calls of the form `new("GraphicalObject", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`boundingBox`: Object of class `"BoundingBox"` describing the position and size of the graphical object.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "GraphicalObject")`: gets the id slot

id<- `signature(object = "GraphicalObject")`: sets the id slot

boundingBox `signature(object = "GraphicalObject")`: gets the `boundingBox` slot

boundingBox<- `signature(object = "GraphicalObject")`: sets the `boundingBox` slot

Author(s)

Michael Lawrence

References

`http://projects.villa-bosch.de/bcb/sbml`

`InitialAssignment-class`

SBML Type "InitialAssignment"

Description

Calculates the value of a symbol when the model is initialized.

Objects from the Class

Objects can be created by calls of the form `new("InitialAssignment", ...)`.

Slots

- `symbol`**:** Object of class `"character"` to which the value is assigned.
- `math`**:** Object of class `"expression"` that evaluates to the assigned value.
- `metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

- **math** `signature(object = "InitialAssignment")`: gets the `math` slot.
- **math<-** `signature(object = "InitialAssignment")`: sets the `math` slot.
- **symbol** `signature(object = "InitialAssignment")`: gets the `symbol` slot.
- **symbol<-** `signature(object = "InitialAssignment")`: sets the `symbol` slot.

Note

Requires libsbml >= 3.0

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

See Also

`AssignmentRule`, which can set a value at any time but cannot set constants.

`KineticLaw-class` *SBML type "KineticLaw"*

Description

Describes the rate of a `Reaction`.

Instantiation

Objects can be created by calls of the form `new("KineticLaw", ...)`.

Slots

- `math`**:** Object of class `"expression"` defining the rate of the reaction.
- `parameters`**:** Object of class `"list"` containing `Parameter`s that may be used in `math`. The names of the list correspond to the IDs of the elements.
- `timeUnits`**:** Object of class `"character"` indicating the units for time.
- `substanceUnits`**:** Object of class `"character"` indicating the units for substance.
- `metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

- **math** `signature(object = "KineticLaw")`: gets the `math` slot
- **math<-** `signature(object = "KineticLaw")`: sets the `math` slot
- **substanceUnits** `signature(object = "KineticLaw")`: gets the `substanceUnits` slot
- **substanceUnits<-** `signature(object = "KineticLaw")`: sets the `substanceUnits` slot
- **timeUnits** `signature(object = "KineticLaw")`: gets the `timeUnits` slot
- **timeUnits<-** `signature(object = "KineticLaw")`: sets the `timeUnits` slot
- **parameters** `signature(object = "KineticLaw")`: gets the `parameters` slot
- **parameters<-** `signature(object = "KineticLaw")`: sets the `parameters` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`Layout-class` *SBML type "Layout"*

Description

Contains the glyphs and other graphical objects that compose an SBML layout. Layouts are not part of the core SBML specification. See the reference for the SBML layout extension specification.

Instantiation

Objects can be created by calls of the form `new("Layout", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`dimensions`: Object of class `"Dimensions"` specifing the size of the layout.

`compartmentGlyphs`: Object of class `"list"` containing the `CompartmentGlyph`s. The names of the list correspond to the IDs of the elements.

`speciesGlyphs`: Object of class `"list"` containing the `SpeciesGlyph`s. The names of the list correspond to the IDs of the elements.

`reactionGlyphs`: Object of class `"list"` containing the `ReactionGlyph`s. The names of the list correspond to the IDs of the elements.

`textGlyphs`: Object of class `"list"` containing the `TextGlyph`s. The names of the list correspond to the IDs of the elements.

`additionalGraphicalObjects`: Object of class `"list"` containing the additional `GraphicalObject`s that are not bound to any model component. The names of the list correspond to the IDs of the elements.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

- **id** `signature(object = "Layout")`: gets the `id` slot
- **id<-** `signature(object = "Layout")`: sets the `id` slot
- **additionalGraphicalObjects** `signature(object = "Layout")`: gets the `additionalGraphicalObjects` slot
- **additionalGraphicalObjects<-** `signature(object = "Layout")`: sets the `additionalGraphicalObjects` slot
- **compartmentGlyphs** `signature(object = "Layout")`: gets the `compartmentGlyphs` slot
- **compartmentGlyphs<-** `signature(object = "Layout")`: sets the `compartmentGlyphs` slot
- **dimensions** `signature(object = "Layout")`: gets the `dimensions` slot
- **dimensions<-** `signature(object = "Layout")`: sets the `dimensions` slot
- **reactionGlyphs** `signature(object = "Layout")`: gets the `reactionGlyphs` slot
- **reactionGlyphs<-** `signature(object = "Layout")`: sets the `reactionGlyphs` slot
- **speciesGlyphs** `signature(object = "Layout")`: gets the `speciesGlyphs` slot
- **speciesGlyphs<-** `signature(object = "Layout")`: sets the `speciesGlyphs` slot
- **textGlyphs** `signature(object = "Layout")`: gets the `textGlyphs` slot
- **textGlyphs<-** `signature(object = "Layout")`: sets the `textGlyphs` slot

Author(s)

Michael Lawrence

References

http://projects.villa-bosch.de/bcb/sbml

`LineSegment-class` *SBML type "LineSegment"*

Description

Describes a simple A-B line.

Instantiation

Objects can be created by calls of the form `new("LineSegment", ...)`.

Slots

- `start`**:** Object of class `"Point"` indicating the start position.
- `end`**:** Object of class `"Point"` indicating the end position.
- `metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

end `signature(x = "LineSegment")`: ...

end<- `signature(object = "LineSegment")`: sets the end slot

start `signature(x = "LineSegment")`: ...

start<- `signature(object = "LineSegment")`: sets the `start` slot

Author(s)

Michael Lawrence

References

`http://projects.villa-bosch.de/bcb/sbml`

`math` *MathML Utilities*

Description

Each of these functions implements a trigonometry function found in the MathML specification but not found in base R. These are all simple wrappers around existing R trig functions.

Usage

```
acot(x)
acoth(x)
acsc(x)
acsch(x)
asec(x)
asech(x)
cot(x)
coth(x)
csc(x)
csch(x)
sec(x)
sech(x)
```

Arguments

`x`	The `numeric` value(s) for the trigonometry operation

Value

A `numeric` vector, the same length as `x`.

Author(s)

Michael Lawrence

`Model-class` *SBML type "Model"*

Description

The central SBML element. Contains the `Species`, `Reactions`, `Compartments` and other components of the model. See the SBML specification, at the reference, for further details.

Instantiation

Objects can be created by calls of the form `new("Model", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`name`: Object of class `"character"` naming this component.

`functionDefinitions`: Object of class `"list"` containing `FunctionDefinition`s. The names of the list correspond to the IDs of the elements.

`unitDefinitions`: Object of class `"list"` containing `UnitDefinition`s. The names of the list correspond to the IDs of the elements.

`compartments`: Object of class `"list"` containing `Compartment`s. The names of the list correspond to the IDs of the elements.

`species`: Object of class `"list"` containing `Species`s. The names of the list correspond to the IDs of the elements.

`parameters`: Object of class `"list"` containing `Parameter`s. The names of the list correspond to the IDs of the elements.

`rules`: Object of class `"list"` containing `Rule`s.

`reactions`: Object of class `"list"` containing `Reaction`s. The names of the list correspond to the IDs of the elements.

`events`: Object of class `"list"` containing `Event`s. The names of the list correspond to the IDs of the elements.

`layouts`: Object of class `"list"` containing `Layout`s. The names of the list correspond to the IDs of the elements.

`speciesTypes`: Object of class `"list"` containing `SpeciesType`s. The names of the list correspond to the IDs of the elements.

`compartmentTypes`: Object of class `"list"` containing `CompartmentType`s. The names of the list correspond to the IDs of the elements.

`constraints`: Object of class `"list"` containing `Constraint`s. The names of the list correspond to the IDs of the elements.

`initialAssignments`**:** Object of class `"list"` containing `InitialAssignments`.

`modelHistory`**:** Object of class `ModelHistory` recording the history of the model.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "Model")`: gets the `id` slot

id<- `signature(object = "Model")`: sets the `id` slot

name `signature(object = "Model")`: gets the `name` slot

name<- `signature(object = "Model")`: sets the `name` slot

compartments `signature(object = "Model")`: gets the `compartments` slot

compartments<- `signature(object = "Model")`: sets the `compartments` slot

events `signature(object = "Model")`: gets the `events` slot

events<- `signature(object = "Model")`: sets the `events` slot

functionDefinitions `signature(object = "Model")`: gets the `functionDefinitions` slot

functionDefinitions<- `signature(object = "Model")`: sets the `functionDefinitions` slot

layouts `signature(object = "Model")`: gets the `layouts` slot

layouts<- `signature(object = "Model")`: sets the `layouts` slot

parameters `signature(object = "Model")`: gets the `parameters` slot

parameters<- `signature(object = "Model")`: sets the `parameters` slot

species `signature(object = "Model")`: gets the `species` slot

species<- `signature(object = "Model")`: sets the `species` slot

reactions `signature(object = "Model")`: gets the `reactions` slot

reactions<- `signature(object = "Model")`: sets the `reactions` slot

rules `signature(object = "Model")`: gets the `rules` slot

rules<- `signature(object = "Model")`: sets the `rules` slot

unitDefinitions `signature(object = "Model")`: gets the `unitDefinitions` slot

unitDefinitions<- `signature(object = "Model")`: sets the `unitDefinitions` slot

compartmentTypes `signature(object = "Model")`: gets the `compartmentTypes` slot

compartmentTypes<- `signature(object = "Model")`: sets the `compartmentTypes` slot

constraints `signature(object = "Model")`: gets the `constraints` slot

constraints<- `signature(object = "Model")`: sets the `constraints` slot

initialAssignments `signature(object = "Model")`: gets the `initialAssignments` slot

initialAssignments<- `signature(object = "Model")`: sets the `initialAssignments` slot

speciesTypes `signature(object = "Model")`: gets the `speciesTypes` slot

speciesTypes<- `signature(object = "Model")`: sets the `speciesTypes`slot

modelHistory `signature(object = "Model")`: gets the `modelHistory` slot

modelHistory<- `signature(object = "Model")`: sets the `modelHistory` slot

stoichiometryMatrix `signature(object = "Model")`: calculates the stoichiometry matrix of the model

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`ModelCreator-class` *SBML Type "ModelCreator"*

Description

Information, such as name, email and organization, about a creator of an SBML model.

Objects from the Class

Objects can be created by calls of the form `new("ModelCreator", ...)`.

Slots

`familyName`**:** Object of class `"character"` specifying the family name of the creator.

`givenName`**:** Object of class `"character"` specifying the given name of the creator.

`email`**:** Object of class `"character"` specifying the email address of the creator.

`organization`**:** Object of class `"character"` specifying the name of the organization to which the creator belongs.

Methods

email `signature(object = "ModelCreator")`: gets the `email` slot.

email<- `signature(object = "ModelCreator")`: sets the `email<-` slot.

familyName `signature(object = "ModelCreator")`: gets the `familyName` slot.

familyName<- `signature(object = "ModelCreator")`: sets the `familyName<-` slot.

givenName `signature(object = "ModelCreator")`: gets the `givenName` slot.

givenName<- `signature(object = "ModelCreator")`: sets the `givenName<-` slot.

organization `signature(object = "ModelCreator")`: gets the `organization` slot.

organization<- `signature(object = "ModelCreator")`: sets the `organization<-` slot.

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`ModelHistory-class` *SBML Type "ModelHistory"*

Description

Stores the history of an SBML model, including the created/modified dates and the creators.

Objects from the Class

Objects can be created by calls of the form `new("ModelHistory", ...)`.

Slots

`createdDate`**:** Object of class `"character"` representing the date/time of creation, in W3CDTF format: `YYYY-MM-DDThh:mm:ssTZD`, e.g. "1997-07-16T19:20:30+01:00".

`modifiedDate`**:** Object of class `"character"` representing the date/time of last modification, in W3CDTF format: `YYYY-MM-DDThh:mm:ssTZD`, e.g. "1997-07-16T19:20:30+01:00".

`creators`**:** Object of class `"list"` of instances of `ModelCreator`, one for each creator of the model.

Methods

createdDate `signature(object = "ModelHistory")`: get the `createdDate` slot.

createdDate<- `signature(object = "ModelHistory", value = "character")`: Set the `createdDate` slot to a correctly formatted string.

createdDate<- `signature(object = "ModelHistory", value = "POSIXt")`: Set the `createdDate` slot with a `POSIXt` instance, obtained e.g. from `Sys.time`.

creators `signature(object = "ModelHistory")`: gets the `creators` slot.

creators<- `signature(object = "ModelHistory")`: sets the `creators` slot.

modifiedDate `signature(object = "ModelHistory")`: get the `modifiedDate` slot.

modifiedDate<- `signature(object = "ModelHistory", value = "character")`: Set the `modifiedDate` slot to a correctly formatted string.

modifiedDate<- `signature(object = "ModelHistory", value = "POSIXt")`: Set the `modifiedDate` slot with a `POSIXt` instance, obtained e.g. from `Sys.time`.

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`ModifierSpeciesReference-class`

SBML type "ModifierSpeciesReference"

Description

Identifies a `Species` that modifies the parent `Reaction`.

Instantiation

Objects can be created by calls of the form `new("ModifierSpeciesReference", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`species`: Object of class `"character"` identifying the `Species` being referenced.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SimpleSpeciesReference"`, directly. Class `"SBase"`, by class "SimpleSpeciesReference", distance 2.

Methods

No methods defined with class "ModifierSpeciesReference" in the signature.

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`Parameter-class` *SBML type "Parameter"*

Description

Declares a variable to be used in a mathematical expression.

Instantiation

Objects can be created by calls of the form `new("Parameter", ...)`.

Slots

`id`**:** Object of class `"character"` uniquely identifying this component.

`name`**:** Object of class `"character"` naming this component.

`value`**:** Object of class `"numeric"` specifying the initial value.

`units`**:** Object of class `"character"` identifying the units.

`constant`**:** Object of class `"logical"` indicating whether the value of this parameter is constant.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "Parameter")`: gets the `id` slot

id<- `signature(object = "Parameter")`: sets the `id` slot

name `signature(object = "Parameter")`: gets the `name` slot

name<- `signature(object = "Parameter")`: sets the `name` slot

units `signature(object = "Parameter")`: gets the `units` slot

units<- `signature(object = "Parameter")`: sets the `units` slot

constant `signature(object = "Parameter")`: gets the `constant` slot

constant<- `signature(object = "Parameter")`: sets the `constant` slot

value `signature(object = "Parameter")`: gets the `value` slot

value<- `signature(object = "Parameter")`: sets the `value` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`ParameterRule-class` *SBML type "ParameterRule"*

Description

Obsolete rule that controls the value of a `Parameter`.

Instantiation

Objects can be created by calls of the form `new("ParameterRule", ...)`.

Slots

`name`: Object of class `"character"` naming this component.

`units`: Object of class `"character"` identifying the units of the assigned value.

`variable`: Object of class `"character"`, ignored.

`type`: Object of class `"character"`, deprecated.

`math`: Object of class `"expression"` specifying the equation.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"AssignmentRule"`, directly. Class `"Rule"`, by class "AssignmentRule", distance 2. Class `"SBase"`, by class "AssignmentRule", distance 3.

Methods

name `signature(object = "Parameter")`: gets the name slot

name<- `signature(object = "Parameter")`: sets the name slot

units `signature(object = "Parameter")`: gets the `units` slot

units<- `signature(object = "Parameter")`: sets the `units` slot

variable `signature(object = "Parameter")`: gets the `variable` slot

variable<- `signature(object = "Parameter")`: sets the `variable` slot

type `signature(object = "Parameter")`: gets the `type` slot

type<- `signature(object = "Parameter")`: sets the `type` slot

math `signature(object = "Parameter")`: gets the `math` slot

math<- `signature(object = "Parameter")`: sets the `math` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`Point-class` *SBML type "Point"*

Description

Specifies a position in 3D space.

Instantiation

Objects can be created by calls of the form `new("Point", ...)`.

Slots

`x`: Object of class `"numeric"` indicating the X coordinate

`y`: Object of class `"numeric"` indicating the Y coordinate

`z`: Object of class `"numeric"` indicating the Z coordinate

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

- **x** `signature(object = "Point")`: gets the x slot
- **x<-** `signature(object = "Point")`: sets the x slot
- **y** `signature(object = "Point")`: gets the y slot
- **y<-** `signature(object = "Point")`: sets the y slot
- **z** `signature(object = "Point")`: gets the z slot
- **z<-** `signature(object = "Point")`: sets the z slot

Author(s)

Michael Lawrence

References

http://projects.villa-bosch.de/bcb/sbml

`RateRule-class` *SBML type "RateRule"*

Description

An equation that describes the rate of change in the quantity of a `Species`, the size of a `Compartment` or the value of a `Parameter`.

Instantiation

Objects can be created by calls of the form `new("RateRule", ...)`.

Slots

- **`variable`:** Object of class `"character"` naming the variable (the id of a `Species`, `Compartment` or `Parameter`) being described.
- **`math`:** Object of class `"expression"` specifying the equation.
- **`metaId`:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- **`notes`:** Object of class `"character"` containing user-readable XHTML notes about an element.
- **`annotation`:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- **`cvTerms`:** Object of class `"list"` containing instances of `CVTerm` associated with this element.
- **`sboTerm`:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"Rule"`, directly. Class `"SBase"`, by class "Rule", distance 2.

Methods

variable `signature(object = "RateRule")`: gets the `variable` slot

variable<- `signature(object = "RateRule")`: sets the `variable` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`Reaction-class` *SBML type "Reaction"*

Description

Any transformation, transportation or binding process that changes the quantity of one or more `Species`.

Instantiation

Objects can be created by calls of the form `new("Reaction", ...)`.

Slots

- `id`: Object of class `"character"` uniquely identifying this component.
- `name`: Object of class `"character"` naming this component.
- `reactants`: Object of class `"list"` containing `SpeciesReferences` specifying the `Species` that are reactants for this reaction. The names of the list correspond to the IDs of the species.
- `products`: Object of class `"list"` containing `SpeciesReferences` specifying the `Species` that are products for this reaction. The names of the list correspond to the IDs of the species.
- `modifiers`: Object of class `"list"` containing `ModifierSpeciesReferences` specifying the `Species` that are modifiers for this reaction. The names of the list correspond to the IDs of the species.
- `kineticLaw`: Object of class `"KineticLaw"` that dynamically defines the rate of the reaction.
- `reversible`: Object of class `"logical"` indicating whether the direction of this reaction is reversible.
- `fast`: Object of class `"logical"` indicating whether this reaction should be considered instantaneous relative to non-fast reactions.
- `metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`: Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`SBase`", directly.

Methods

id `signature(object = "Reaction")`: gets the `id` slot

id<- `signature(object = "Reaction")`: sets the `id` slot

name `signature(object = "Reaction")`: gets the `name` slot

name<- `signature(object = "Reaction")`: sets the `name` slot

fast `signature(object = "Reaction")`: gets the `fast` slot

fast<- `signature(object = "Reaction")`: sets the `fast` slot

kineticLaw `signature(object = "Reaction")`: gets the `kineticLaw` slot

kineticLaw<- `signature(object = "Reaction")`: sets the `kineticLaw` slot

modifiers `signature(object = "Reaction")`: gets the `modifiers` slot

modifiers<- `signature(object = "Reaction")`: sets the `modifiers` slot

products `signature(object = "Reaction")`: gets the `products` slot

products<- `signature(object = "Reaction")`: sets the `products` slot

reactants `signature(object = "Reaction")`: gets the `reactants` slot

reactants<- `signature(object = "Reaction")`: sets the `reactants` slot

reversible `signature(object = "Reaction")`: gets the `reversible` slot

reversible<- `signature(object = "Reaction")`: sets the `reversible` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`ReactionGlyph-class` *SBML type "ReactionGlyph"*

Description

A glyph representing a `Reaction` in the SBML layout.

Instantiation

Objects can be created by calls of the form `new("ReactionGlyph", ...)`.

Slots

- `reaction`: Object of class `"character"` identifying the reaction represented by this glyph.
- `glyphCurve`: Object of class `"Curve"` describing this glyph as a curve (optional).
- `speciesReferenceGlyphs`: Object of class `"list"` containing `SpeciesReferenceGlyphs` that represent the `SpeciesReferences` of the underlying `Reaction`. The names of the list correspond to the IDs of the elements.
- `id`: Object of class `"character"` uniquely identifying this component.
- `boundingBox`: Object of class `"BoundingBox"` describing the position and size of the graphical object.
- `metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`: Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`GraphicalObject`", directly. Class "`SBase`", by class "GraphicalObject", distance 2.

Methods

- **glyphCurve** `signature(expr = "ReactionGlyph")`: gets the `glyphCurve` slot
- **glyphCurve<-** `signature(object = "ReactionGlyph")`: sets the `glyphCurve` slot
- **reaction** `signature(object = "ReactionGlyph")`: gets the `reaction` slot
- **reaction<-** `signature(object = "ReactionGlyph")`: sets the `reaction` slot
- **speciesReferenceGlyphs** `signature(object = "ReactionGlyph")`: gets the `speciesReferenceGlyphs` slot
- **speciesReferenceGlyphs<-** `signature(object = "ReactionGlyph")`: sets the `speciesReferenceGlyphs` slot

Author(s)

Michael Lawrence

References

`http://projects.villa-bosch.de/bcb/sbml`

`Rule-class` *SBML type "Rule"*

Description

A mathematical equation.

Instantiation

A virtual Class: No objects may be created from it.

Slots

- `math`**:** Object of class `"expression"` specifying the equation.
- `metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

- **math** `signature(object = "Rule")`: gets the `math` slot
- **math<-** `signature(object = "Rule")`: sets the `math` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`SBase-class` *SBML type "SBase"*

Description

The abstract type from which all other SBML types are derived.

Instantiation

A virtual Class: No objects may be created from it.

Slots

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Methods

annotation `signature(object = "SBase")`: gets the `annotation` slot

annotation<- `signature(object = "SBase")`: sets the `annotation` slot

metaId `signature(object = "SBase")`: gets the `metaId` slot

metaId<- `signature(object = "SBase")`: sets the `metaId` slot

notes `signature(object = "SBase")`: gets the `notes` slot

notes<- `signature(object = "SBase")`: sets the `notes`slot

cvTerms `signature(object = "SBase")`: gets the `cvTerms` slot.

cvTerms<- `signature(object = "SBase")`: sets the `cvTerms` slot.

sboTerm `signature(object = "SBase")`: gets the `sboTerm` slot.

sboTerm<- `signature(object = "SBase")`: sets the `sboTerm` slot.

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`SBML import` *Read in an SBML file (start here)*

Description

Read an SBML file into R.

Usage

```
rsbml_read(filename, text, dom = TRUE, strict = FALSE, schema = FALSE,
                    consistency = TRUE)
```

Arguments

`filename`	the name of the SBML file to parse
`text`	a string of SBML text to parse (instead of file)
`dom`	whether to convert directly to the S4 DOM (`TRUE`, the default) or leave as the internal `SBMLDocument`.
`strict`	whether to report warnings in addition to errors or not (`FALSE`, the default).
`schema`	whether to perform XML schema validation
`consistency`	whether to perform consistency checks; recommended but might cause performance deficiencies.

Value

a `SBML` object, or a `SBMLDocument` if dom is `FALSE`.

Author(s)

Michael Lawrence

Examples

```
# Read an SBML file
file <- system.file("sbml", "GlycolysisLayout.xml", package = "rsbml")
doc <- rsbml_read(file)

# Read an SBML string
string <- paste(readLines(file),collapse="\n")
doc <- rsbml_read(text = string)
```

`SBML-class` *SBML type "SBML"*

Description

The root element of an SBML document. An actual SBML `Model` may be retrieved from an instance of this class.

Instantiation

Objects can be created by calls of the form `new("SBML", ...)`.

Slots

`level`**:** Object of class `"integer"` indicating the level of the SBML standard (currently at 2).

`ver`**:** Object of class `"integer"` indicating the version of the level (currently at 2 for level 2).

`model`**:** Object of class `"Model"` the SBML model itself.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

coerce `signature(from = "SBMLDocument", to = "SBML")`: constructs the S4 object model from a low-level libsbml document.

coerce `signature(from = "SBML", to = "SBMLDocument")`: converts the S4 object model to a low-level libsbml document.

coerce `signature(from = "SBML", to = "graph")`: converts the S4 object model to a graph.

level `signature(object = "SBML")`: gets the `level` slot

level<- `signature(object = "SBML")`: sets the `level` slot

model `signature(object = "SBML")`: gets the `model` slot

model<- `signature(object = "SBML")`: sets the `model` slot

rsbml_doc `signature(model = "SBML")`: converts the S4 object model to a low-level libsbml document.

rsbml_write `signature(object = "SBML")`: writes this document to a file as SBML.

rsbml_xml `signature(object = "SBML")`: converts this document to a string as SBML.

rsbml_graph `signature(object = "SBML")`: converts this document to a graph object.

rsbml_check `signature(object = "SBML")`: perform consistency checks, see `rsbml_check`.

simulate `signature(object = "SBML")`: converts this document to an internal `SBMLDocument` and calls `simulate` on it.

ver `signature(object = "SBML")`: gets the `ver` slot

ver<- `signature(object = "SBML")`: sets the `ver` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

Examples

```
# Get a DOM
dom <- rsbml_read(system.file("sbml", "GlycolysisLayout.xml", package = "rsbml"))

# Get the species ID's
sapply(species(model(dom)), id)

# Convert DOM back to a low-level document for checking
doc <- rsbml_doc(dom)
rsbml_check(doc)

# Write a DOM to a file
## Not run: rsbml_write(dom, "my.xml")
```

`SBMLDocument-class` *"SBMLDocument" from libsbml*

Description

Low-level libsbml document structure.

Instantiation

A virtual Class: No objects may be created from it.

Extends

Class "`oldClass`", directly.

Methods

rsbml_check `signature(object = "SBMLDocument")`: `rsbml_check(object, strict = FALSE, consistency = TRUE)`: Check for problems with the document and signal R conditions if any errors are detected. If `strict` is `TRUE`, libsbml warnings will be emitted as R warnings (these are often too pedantic). If `consistency` is also `TRUE`, reports problems regarding internal model inconsistencies. This can be time consuming.

rsbml_dom `signature(doc = "SBMLDocument")`: Constructs an S4 object model from a libsbml document.

rsbml_graph `signature(doc = "SBMLDocument")`: Converts a libsbml document to a `graph`.

rsbml_problems `signature(object = "SBMLDocument")`: reports problems encountered during parsing and/or validation.

rsbml_write `signature(object = "SBMLDocument")`: writes this document to a file as SBML.

rsbml_xml `signature(object = "SBMLDocument")`: converts this document to a string as SBML.

simulate `signature(object = "SBMLDocument")`: `simulate(object, nsim = 10, seed, ...)`: a shortcut for simulating the model in this document using the SBML ODE Solver library. Arguments in ... should match slots of `SOSProtocol`. See `simulate` for more details.

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

Examples

```
  # Read a document into an R DOM
  dom <- rsbml_read(system.file("sbml", "GlycolysisLayout.xml", package
= "rsbml"))

  # Convert to a graph
  graph <- rsbml_graph(dom)

  # Write it out to a file
  ## Not run: rsbml_write(dom, "my.xml")

  # Or convert it to a string of XML
  rsbml_xml(dom)

  # Read into external libsbml data structure
  doc <- rsbml_read(system.file("sbml", "GlycolysisLayout.xml", package
= "rsbml"), dom = FALSE)

  # Convert it explicitly to an S4 DOM
  dom <- rsbml_dom(doc)
```

`SBMLProblem-class` *SBMLProblem*

Description

Represents an exception thrown during SBML parsing.

Details

There are trivial subclasses for fatal errors (`SBMLFatal`), recoverable errors (`SBMLError`), warnings (`SBMLWarning`) and informational messages (`SBMLInfo`). Errors become R `error` `condition`s, warnings become R `warning` conditions and messages are output via `message`.

Slots

`line`: The `"numeric"` line number in the SBML file where the problem was detected.

`column`: Object of class `"numeric"` column number in the SBML file where the problem was detected.

`msg`: Object of class `"character"`, a human-readable description of the problem.

Methods

.condition `signature(object = "SBMLProblem")`: constructs a `condition` object representing the exception.

Author(s)

Michael Lawrence

See Also

`SBMLProblems`, a container for instances of this class.

`SBMLProblems-class` *SBMLProblems*

Description

A class representing errors encountered during parsing of SBML.

Slots

`fatals`: A list of `SBMLFatal` instances.

`errors`: A list of `SBMLError` instances.

`warnings`: A list of `SBMLWarning` instances.

`infos`: A list of `SBMLInfo` instances.

Methods

- **.throw** `signature(object = "SBMLProblems")`: Throws each `SBMLProblem` in this object.
- **errors** `signature(object = "SBMLProblems")`: Gets the `errors` slot.
- **fatals** `signature(object = "SBMLProblems")`: Gets the `fatals` slot.
- **infos** `signature(object = "SBMLProblems")`: Gets the `infos` slot.
- **warns** `signature(object = "SBMLProblems")`: Gets the `warns` slot.

Author(s)

Michael Lawrence

See Also

The `rsbml_problems` function for obtaining an instance of this class describing any problems encountered during parsing.

`SimpleSpeciesReference-class`

SBML type "SimpleSpeciesReference"

Description

Base class for bindings between a `Species` and a `Reaction`.

Instantiation

Objects can be created by calls of the form `new("SimpleSpeciesReference", ...)`.

Slots

- `id`: Object of class `"character"` uniquely identifying this component.
- `species`: Object of class `"character"` identifying the `Species` being referenced.
- `metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`: Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "SpeciesGlyph")`: gets the `id` slot

id<- `signature(object = "SpeciesGlyph")`: sets the `id` slot

species `signature(object = "SpeciesGlyph")`: gets the `species` slot

species<- `signature(object = "SpeciesGlyph")`: sets the `species` slot

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

`SOSDesign-class` *SOSDesign*

Description

Specifies the reaction names and their parameter settings for each run in a batch experiment. It extends `matrix`; each column corresponds to a parameter in the model and each row should hold the parameter settings for one run of the experiment.

Details

It is often desirable to explore the state space of a model by adjusting its initial parameter settings. One could do this by modifying the model itself for each experiment, but this class aims to provide a more convenient and systematic means of running experiments in batch, over a range of parameter settings. The results of the experiment will then contain the output from each run, which may then be compared.

The design is specified as a matrix, and each column in the matrix should correspond to a parameter defined in an SBML model. The column names should identify the parameters. These are not to be confused with the simulation parameters specified in `SOSProtocol`, which control how the simulation is executed. These should be and are designed to be kept constant across the runs.

There are two different types of parameters: global and local (reaction) parameters. Global parameters may correspond to a `Species` quantity, `Compartment` size, or model-level `Parameter` value. These should be identified in the column names by the `id` of the corresponding SBML element. The element in the `reactions` slot for one of these parameters should be the empty string.

The second type of parameter specifies the value of a `Parameter` element within the `KineticLaw` of a reaction. These should be named by the `id` of the `Parameter`. They also should be namespaced by the containing `Reaction id`, which is stored in the corresponding element of the `reactions` slot.

Objects from the Class

Objects can be created by calls of the form `new("SOSDesign", data, nrow, ncol, byrow, dimnames, ...)`. This is the same as initializing a `matrix`.

Slots

`.Data`: Object of class `"matrix"`, holding the parameter settings.

`reactions`: Object of class `"character"` of length the number of columns, holding the reaction IDs for parameters local to a reaction (i.e. `KineticLaw` `Parameter`s). For global parameters, the corresponding value should be the empty string.

Extends

Class `"matrix"`, from data part. Class `"ExperimentDesign"`, directly. Class `"array"`, by class "matrix", distance 2. Class `"structure"`, by class "matrix", distance 3. Class `"vector"`, by class "matrix", distance 4, with explicit coerce.

Methods

reactions `signature(object = "SOSDesign")`: gets the `reactions` slot.

reactions<- `signature(object = "SOSDesign")`: sets the `reactions` slot.

Author(s)

Michael Lawrence

References

See `http://www.tbi.univie.ac.at/~raim/odeSolver/` for more information on the SBML ODE Solver library.

See Also

`SOSExperiment`, the container of this class, for configuring and running a simulation.

`SOSExperiment-class` *SOS Experiment*

Description

Implementation of `Experiment` for simulating SBML models using the SOS: (S)BML (O)DE (S)olver library.

Details

The general workflow for running a simulation:

1. Create or import an SBML DOM.
2. Customize the model, for example by adding perturbation `Event`s.
3. Wrap the SBML DOM in a `SOSSubject`, e.g. `new("SOSSubject", dom)`.
4. Optionally construct a `SOSDesign` for running the experiment in batch over several sets of model parameter settings.
5. Optionally construct a `SOSProtocol` for specifying the time points and other parameters controlling the simulation.
6. Construct an instance of this class that groups the subject, design and protocol.

7. Run `simulate` on the `SOSExperiment`, optionally specifying the number of iterations and the random seed.
8. Analyze the returned `SOSResult`, perhaps starting by converting it to a time series with `as.ts` and making some plots.

Objects from the Class

Objects can be created by calls of the form `new("SOSExperiment", ...)`.

Slots

`protocol`**:** Object of class `SOSProtocol`, where the simulation parameters are specified.

`design`**:** Object of class `SOSDesign`, specifying model parameters for each run of a batch experiment.

`subject`**:** Object of class `SOSSubject`, containing the `Model` to be simulated.

`result`**:** Object of class `SOSResult` containing the result of the simulation.

Extends

Class `Experiment`, directly.

Methods

simulate `signature(object = "SOSExperiment")`: `simulate(object, nsim = 10, seed, ...)`: Simulates the SBML document in the `subject` slot according to the design points in `design` and parameters in `protocol` for `nsim` iterations, using `seed` as the random seed. Returns an instance of `SOSExperiment`, which now should include a `SOSResult` for analysis.

Author(s)

Michael Lawrence

References

See `http://www.tbi.univie.ac.at/~raim/odeSolver/` for more information on the SBML ODE Solver library.

See Also

The `simulate` method on `SBMLDocument` is a shortcut, but most users will probably find the above approach most useful.

`SOSProtocol-class` *SOSProtocol*

Description

Holds the parameters controlling the execution of the simulation using the SBML ODE Solver library.

Details

Most users will probably set only the `times` slot, either directly or through the `timeStep` slot and the `nsim` parameter to `simulate`.

Objects from the Class

Objects can be created by calls of the form `new("SOSProtocol", ...)`. Each argument in `...` should correspond to one of the slots described below.

Slots

- **`times`:** A `"numeric"` vector indicating the time points at which to evaluate the model. Defaults to `tail(seq(0, by = timeStep, length.out = nsim + 1), -1)`. The model is always evaluated at t = 0. This slot is ignored when `indefinite` (below) is `TRUE`.
- **`timeStep`:** A scalar `"numeric"` value, giving the length in time between model evaluations. This is used when calculating the default value of `times`, above, but is otherwise only relevant when the `indefinite` slot, below, is `TRUE`. Defaults to 1.
- **`indefinite`:** A scalar `"logical"`, indicating whether the simulation should run indefinitely, i.e. until one of the stopping conditions is met. See `haltOnEvent` and `haltOnSteadyState` below. Defaults to `FALSE`.
- **`atol`:** Scalar `"numeric"`, the absolute tolerance in integration error. Defaults to `1e-18`.
- **`rtol`:** Scalar `"numeric"`, the relative tolerance in integration error. Defaults to `1e-10`.
- **`maxStep`:** Sclar `"numeric"`, the maximum number of steps for integration. Not to be confused with `timeStep`, etc, above, which control the simulation time points. Defaults to `10000`.
- **`odeMethod`:** Scalar `"character"` naming the method for solving ODEs. Either `"bdf"` (the default) or `"adams-moulton"`.
- **`iterMethod`:** Scalar `"character"`, naming the iteration method used by the ODE solver, either `"newton"` (the default) or `"functional"`.
- **`maxOrder`:** Scalar `"numeric"` indicating maximum order for the ODE solver. Defaults to `5`.
- **`sensMethod`:** Scalar `"character"` naming the method for sensitivity analysis. One of `"none"` (the default and currently the only valid option), `"simultaneous"`, `"staggered"` or `"staggered1"`.
- **`haltOnEvent`:** Scalar `"logical"` indicating whether the simulation should halt when the model emits an `Event`. This allows the model to stop the simulation when some state is reached. Defaults to `FALSE`.
- **`haltOnSteadyState`:** Scalar `"logical"`, indicating whether to halt when a steady state is detected. Defaults to `FALSE`.
- **`useJacobian`:** Scalar `"logical"` indicating whether to use Jacobian ASTs (`TRUE`, the default) or the internal approximation in the CVODES library.
- **`storeResults`:** Scalar `"logical"` indicating whether to store the entire time course (`TRUE`, the default) or just the last time point. Just for performance.

Extends

Class "`ExperimentProtocol`", directly.

Methods

No methods defined with class "SOSProtocol" in the signature.

Author(s)

Michael Lawrence

References

See `http://www.tbi.univie.ac.at/~raim/odeSolver/` for more information on the SBML ODE Solver library.

See Also

The `SOSExperiment` class, which contains a `SOSProtocol` instance, for setting up and running a simulation.

`SOSResult-class` *SOSResult*

Description

A result from simulating an `SOSExperiment`. Contains the time course for each of the model variables: the `Species` quantities, `Compartment` sizes, `Parameter` values, and `Reaction` rates.

Slots

`data`**:** A `"data.frame"` containing the time course data. Each row contains the value at a single time point for a single time course. Has the following columns:

- `sample` A factor, the run number, only exists if there were multiple runs, see `SOSDesign`.
- `type` A factor, the SBML element type for the time course, e.g. `"species"`.
- `id` A factor, the `id` of the SBML element for the time course.
- `time` The numeric time value for the time point.
- `value` The actual numeric value for the time course at that time.

`sens`**:** A `"matrix"` with results from sensitivity analysis, not yet supported.

Extends

Class "`ExperimentResult`", directly.

Methods

- **as.ts** `signature(x = "SOSResult")`: converts this object to a time course object of class `ts`. This allows analysis of the results with existing R infrastructure for time course analysis.
- **compartments** `signature(object = "SOSResult")`: returns a subset containing only the `Compartment` size courses.
- **parameters** `signature(object = "SOSResult")`: returns a subset containing only the global `Parameter` value courses.
- **reactions** `signature(object = "SOSResult")`: returns a subset containing only the `Reaction` rate courses.
- **species** `signature(object = "SOSResult")`: returns a subset containing only the `Species` quantity courses.

Author(s)

Michael Lawrence

References

See `http://www.tbi.univie.ac.at/~raim/odeSolver/` for more information on the SBML ODE Solver library.

See Also

`SOSExperiment` for running a simulation and obtaining an instance of this class.

`SOSSubject-class` *SOSSubject*

Description

This just marks an `SBML` object as being a valid subject for simulation using the SBML ODE Solver library.

Objects from the Class

Normally created from a `SBML` with: `new("SOSSubject", model)`.

Extends

Class "`ExperimentSubject`", directly. Class "`SBML`", directly. Class "`SBase`", by class "SBML", distance 2. Class "`Describable`", by class "SBML", distance 3.

Author(s)

Michael Lawrence

References

See `http://www.tbi.univie.ac.at/~raim/odeSolver/` for more information on the SBML ODE Solver library.

See Also

`SOSExperiment` for running a simulation on a `SOSSubject`.

`Species-class` *SBML type "Species"*

Description

A participant in an SBML model.

Instantiation

Objects can be created by calls of the form `new("Species", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`name`: Object of class `"character"` naming this component.

`compartment`: Object of class `"character"` identifying the compartment in which this species is located.

`initialAmount`: Object of class `"numeric"` indicating the initial amount for this species (mutually exclusive with `initialConcentration`).

`initialConcentration`: Object of class `"numeric"` indicating the initial concentration for this species (mutually exclusive with `initialAmount`).

`substanceUnits`: Object of class `"character"` identifying the units for the amount of this species or the numerator of the concentration.

`spatialSizeUnits`: Object of class `"character"` identifying the units for the denominator of the species concentration.

`hasOnlySubstanceUnits`: Object of class `"logical"` indicating whether the quantity of this species is specified as an amount or a concentration.

`boundaryCondition`: Object of class `"logical"`. If `TRUE`, indicates that the quantity of this species cannot be changed by a reaction.

`charge`: Object of class `"integer"` indicating the electrical charge of this species.

`constant`: Object of class `"logical"` indicating whether the quantity of this species can change.

`units`: Object of class `"character"`, deprecated.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`SBase`", directly.

Methods

id `signature(object = "Species")`: gets the `id` slot

id<- `signature(object = "Species")`: sets the `id` slot

name `signature(object = "Species")`: gets the `name` slot

name<- `signature(object = "Species")`: sets the `name` slot

boundaryCondition `signature(object = "Species")`: gets the `boundaryCondition` slot

boundaryCondition<- `signature(object = "Species")`: sets the `boundaryCondition` slot

charge `signature(object = "Species")`: gets the `charge` slot

charge<- `signature(object = "Species")`: sets the `charge` slot

compartment `signature(object = "Species")`: gets the `compartment` slot

compartment<- `signature(object = "Species")`: sets the `compartment` slot

constant `signature(object = "Species")`: gets the `constant` slot

constant<- `signature(object = "Species")`: sets the `constant` slot

units `signature(object = "Species")`: gets the `constant` slot

units<- `signature(object = "Species")`: sets the `constant` slot

hasOnlySubstanceUnits `signature(object = "Species")`: gets the `hasOnlySubstanceUnits` slot

hasOnlySubstanceUnits<- `signature(object = "Species")`: sets the `hasOnlySubstanceUnits` slot

initialAmount `signature(object = "Species")`: gets the `initialAmount` slot

initialAmount<- `signature(object = "Species")`: sets the `initialAmount` slot

initialConcentration `signature(object = "Species")`: gets the `initialConcentration` slot

initialConcentration<- `signature(object = "Species")`: sets the `initialConcentration` slot

spatialSizeUnits `signature(object = "Species")`: gets the `spatialSizeUnits` slot

spatialSizeUnits<- `signature(object = "Species")`: sets the `spatialSizeUnits` slot

substanceUnits `signature(object = "Species")`: gets the `substanceUnits` slot

substanceUnits<- `signature(object = "Species")`: sets the `substanceUnits` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`SpeciesConcentrationRule-class`

SBML type "SpeciesConcentrationRule"

Description

Obsolete type of rule that describes the concentration of `Species`.

Instantiation

Objects can be created by calls of the form `new("SpeciesConcentrationRule", ...)`.

Slots

`species`: Object of class `"character"` identifying the `Species`.

`variable`: Object of class `"character"`, ignored.

`type`: Object of class `"character"`, deprecated.

`math`: Object of class `"expression"` specifying the equation.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"AssignmentRule"`, directly. Class `"Rule"`, by class "AssignmentRule", distance 2. Class `"SBase"`, by class "AssignmentRule", distance 3.

Methods

species `signature(object = "SpeciesConcentrationRule")`: gets the species slot

species<- `signature(object = "SpeciesConcentrationRule")`: sets the species slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`SpeciesGlyph-class` *SBML type "SpeciesGlyph"*

Description

A glyph representing a `Species` in an SBML layout.

Instantiation

Objects can be created by calls of the form `new("SpeciesGlyph", ...)`.

Slots

`species`**:** Object of class `"character"` identifying the species this glyph represents.

`id`**:** Object of class `"character"` uniquely identifying this component.

`boundingBox`**:** Object of class `"BoundingBox"` describing the position and size of the graphical object.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`GraphicalObject`", directly. Class "`SBase`", by class "GraphicalObject", distance 2.

Methods

species `signature(object = "SpeciesGlyph")`: gets the `species` slot

species<- `signature(object = "SpeciesGlyph")`: sets the `species` slot

Author(s)

Michael Lawrence

References

http://projects.villa-bosch.de/bcb/sbml

`SpeciesReference-class`

SBML type "SpeciesReference"

Description

Binds a reactant or product `Species` to a `Reaction`.

Instantiation

Objects can be created by calls of the form `new("SpeciesReference", ...)`.

Slots

- `stoichiometry`**:** Object of class `"numeric"` indicating the (static) stoichiometric coefficient.
- `stoichiometryMath`**:** Object of class `"StoichiometryMath"` that dynamically calculates the stoichiometric coefficient.
- `id`**:** Object of class `"character"` uniquely identifying this component.
- `species`**:** Object of class `"character"` identifying the `Species` being referenced.
- `metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`SimpleSpeciesReference`", directly. Class "`SBase`", by class "SimpleSpeciesReference", distance 2.

Methods

- **stoichiometry** `signature(object = "SpeciesReference")`: gets the `stoichiometry` slot
- **stoichiometry<-** `signature(object = "SpeciesReference")`: sets the `stoichiometry` slot
- **stoichiometryMath** `signature(object = "SpeciesReference")`: gets the `stoichiometryMath` slot
- **stoichiometryMath<-** `signature(object = "SpeciesReference")`: sets the `stoichiometryMath` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`SpeciesReferenceGlyph-class`

SBML type "SpeciesReferenceGlyph"

Description

A glyph representing a `SpeciesReference` in an SBML layout.

Instantiation

Objects can be created by calls of the form `new("SpeciesReferenceGlyph", ...)`.

Slots

`speciesGlyph`**:** Object of class `"character"` identifying the `SpeciesGlyph` representing the `Species` that is referenced by the underlying `SpeciesReference`.

`speciesReference`**:** Object of class `"character"` identifying the `linkS4class{SpeciesReference}` represented by this glyph.

`role`**:** Object of class `"character"` indicating how this glyph should represent the "role" of the underlying `SpeciesReference`.

`glyphCurve`**:** Object of class `"Curve"` describing this glyph as a curve (optional).

`id`**:** Object of class `"character"` uniquely identifying this component.

`boundingBox`**:** Object of class `"BoundingBox"` describing the position and size of the graphical object.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"GraphicalObject"`, directly. Class `"SBase"`, by class "GraphicalObject", distance 2.

Methods

role `signature(object = "SpeciesReferenceGlyph")`: gets the `role` slot

role<- `signature(object = "SpeciesReferenceGlyph")`: sets the `role` slot

speciesGlyph `signature(object = "SpeciesReferenceGlyph")`: gets the `speciesGlyph` slot

speciesGlyph<- `signature(object = "SpeciesReferenceGlyph")`: sets the `speciesGlyph` slot

speciesReference `signature(object = "SpeciesReferenceGlyph")`: gets the `speciesReference` slot

speciesReference<- `signature(object = "SpeciesReferenceGlyph")`: sets the `speciesReference` slot

glyphCurve `signature(expr = "SpeciesReferenceGlyph")`: gets the `glyphCurve` slot

glyphCurve<- `signature(object = "SpeciesReferenceGlyph")`: sets the `glyphCurve` slot

Author(s)

Michael Lawrence

References

`http://projects.villa-bosch.de/bcb/sbml`

`SpeciesType-class` *SBML Type "SpeciesType"*

Description

A `Species` represents a pool of a chemical in a particular `linkS4class{Compartment}`. This element specifies a type of species, that is, the chemical independent of location.

Objects from the Class

Objects can be created by calls of the form `new("SpeciesType", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`name`: Object of class `"character"` naming this component.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

id `signature(object = "SpeciesType")`: gets the `id` slot

id<- `signature(object = "SpeciesType")`: sets the `id` slot

name `signature(object = "SpeciesType")`: gets the `name` slot

name<- `signature(object = "SpeciesType")`: sets the `name` slot

Note

Requires libsbml >= 3.0

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

See Also

`Species`

`StoichiometryMath-class`

SBML type "StoichiometryMath"

Description

Dynamically defines the stoichiometry of a `SpeciesReference`.

Instantiation

Objects can be created by calls of the form `new("StoichiometryMath", ...)`.

Slots

`math`**:** Object of class `"expression"` that evaluates to the stoichiometric coefficient.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

math `signature(object = "StoichiometryMath")`: gets the `math` slot

math<- `signature(object = "StoichiometryMath")`: sets the `math` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`TextGlyph-class` *SBML type "TextGlyph"*

Description

A run of text in an SBML layout.

Instantiation

Objects can be created by calls of the form `new("TextGlyph", ...)`.

Slots

- `graphicalObject`: Object of class `"character"` identifying the `GraphicalObject` that this glyph labels (optional).
- `text`: Object of class `"character"` containing the text shown by the glyph (mutually exclusive with `originOfText`).
- `originOfText`: Object of class `"character"` identifying an SBML component whose name is used as the text (mutually exclusive with `text`).
- `id`: Object of class `"character"` uniquely identifying this component.
- `boundingBox`: Object of class `"BoundingBox"` describing the position and size of the graphical object.
- `metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`: Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"GraphicalObject"`, directly. Class `"SBase"`, by class "GraphicalObject", distance 2.

Methods

graphicalObject `signature(object = "TextGlyph")`: gets the `graphicalObject` slot

graphicalObject<- `signature(object = "TextGlyph")`: sets the `graphicalObject` slot

originOfText `signature(object = "TextGlyph")`: gets the `originOfText` slot

originOfText<- `signature(object = "TextGlyph")`: sets the `originOfText` slot

text `signature(x = "TextGlyph")`: ...

text<- `signature(object = "TextGlyph")`: sets the `text` slot

Author(s)

Michael Lawrence

References

`http://projects.villa-bosch.de/bcb/sbml`

`Trigger-class` *SBML Type "Trigger"*

Description

Expresses when an `Event` should be fired.

Objects from the Class

Objects can be created by calls of the form `new("Trigger", ...)`.

Slots

`math`**:** Object of class `"expression"` that evaluates to `TRUE` when the event should be fired.

`metaId`**:** Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`**:** Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`**:** Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`**:** Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`**:** Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class `"SBase"`, directly.

Methods

math `signature(domain = "Trigger")`: gets the `math` slot.

math<- `signature(object = "Trigger")`: sets the `math` slot.

Note

Requires libsbml >= 3.0

Author(s)

Michael Lawrence

References

http://sbml.org/documents/

See Also

`Event`, the parent of this element.

`Unit-class` *SBML type "Unit"*

Description

A (possibly transformed) reference to a base `UnitKind`. The transformation is of the form: $multiplier * 10^scale * x^exponent + offset$.

Instantiation

Objects can be created by calls of the form `new("Unit", ...)`.

Slots

- `kind`: Object of class `"character"` identifying a an SBML `UnitKind`. For possible values see Table 2 in the SBML specification.
- `exponent`: Object of class `"integer"` indicating the exponent to use in the transformation.
- `unitScale`: Object of class `"integer"` indicating the order of magnitude of the scaling to use in the transformation.
- `multiplier`: Object of class `"numeric"` indicating the factor to use for scaling in the transformation.
- `offset`: Object of class `"numeric"` indicating the amount of constant shift in the transformation.
- `metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.
- `notes`: Object of class `"character"` containing user-readable XHTML notes about an element.
- `annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.
- `cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.
- `sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`SBase`", directly.

Methods

exponent `signature(object = "Unit")`: gets the `exponent` slot

exponent<- `signature(object = "Unit")`: sets the `exponent` slot

kind `signature(object = "Unit")`: gets the `kind` slot

kind<- `signature(object = "Unit")`: sets the `kind` slot

multiplier `signature(object = "Unit")`: gets the `multiplier` slot

multiplier<- `signature(object = "Unit")`: sets the `multiplier` slot

offset `signature(object = "Unit")`: gets the `offset` slot

offset<- `signature(object = "Unit")`: sets the `offset` slot

unitScale `signature(x = "Unit")`: ...

unitScale<- `signature(object = "Unit")`: sets the `unitScale` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

`UnitDefinition-class` *SBML type "UnitDefinition"*

Description

Associates one or more `Unit`s with an ID and name.

Instantiation

Objects can be created by calls of the form `new("UnitDefinition", ...)`.

Slots

`id`: Object of class `"character"` uniquely identifying this component.

`name`: Object of class `"character"` naming this component.

`units`: Object of class `"list"` containing equivalent `Unit`s that are all associated with the same ID and name.

`metaId`: Object of class `"character"` that is an XML ID "described" by an RDF resource. This links an SBML element to an RDF resource. RDF may appear anywhere in an SBML element, but is usually placed inside the `annotation` element.

`notes`: Object of class `"character"` containing user-readable XHTML notes about an element.

`annotation`: Object of class `"character"` containing additional machine-readable information about an element, usually as RDF, such as BioPAX. This is where application-specific data belongs.

`cvTerms`: Object of class `"list"` containing instances of `CVTerm` associated with this element.

`sboTerm`: Object of class `"integer"` identifying a term in the Systems Biology Ontology (SBO).

Extends

Class "`SBase`", directly.

Methods

- **id** `signature(object = "UnitDefinition")`: gets the `id` slot
- **id<-** `signature(object = "UnitDefinition")`: sets the `id` slot
- **name** `signature(object = "UnitDefinition")`: gets the `name` slot
- **name<-** `signature(object = "UnitDefinition")`: sets the `name` slot
- **units** `signature(object = "UnitDefinition")`: gets the `units` slot
- **units<-** `signature(object = "UnitDefinition")`: sets the `units` slot

Author(s)

Michael Lawrence

References

`http://sbml.org/documents/`

Index