

Groups

Michael Hucka

mhucka@caltech.edu

Computing and Mathematical Sciences
California Institute of Technology
Pasadena, California, US

Lucian P. Smith

lpsmith@u.washington.edu

Computing and Mathematical Sciences
California Institute of Technology
Seattle, Washington, US

Version 1, Release 0.4 (Draft)

31 May 2013

This is a draft specification for the SBML Level 3 package called “groups”. It is not a normative document. Please send feedback to the Package Working Group mailing list at sbml-groups@lists.sourceforge.net.

The latest release, past releases, and other materials related to this specification are available at http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/groups

This release of the specification is available at
TBD



Contents

1	Introduction	3
1.1	Proposal corresponding to this package specification	3
1.2	Package dependencies	3
1.3	Document conventions	3
2	Background and context	4
2.1	Prior work	4
3	Package syntax and semantics	5
3.1	Namespace URI and other declarations necessary for using this package	5
3.2	Primitive data types	5
3.2.1	Type GroupKind	5
3.3	The Group class	5
3.3.1	The id and name attributes	6
3.3.2	The kind attribute	6
3.4	The ListOfMembers class	7
3.5	The Member class	7
3.5.1	The id and name attributes	7
3.5.2	The idRef attribute	7
3.5.3	The metaIdRef attribute	8
3.5.4	Members referencing other Groups and Members	8
3.5.5	Members referencing other namespaces	8
3.5.6	Example	8
3.6	The ListOfMemberConstraints class	9
3.6.1	The id and name attributes	9
3.6.2	The membersShareType attribute	9
3.7	The MemberConstraint class	10
3.7.1	The id and name attributes	10
3.7.2	The distinctAttribute attribute	10
3.7.3	The identicalAttribute attribute	10
3.7.4	Example	10
3.8	The extended Model class	11
3.8.1	The list of groups	11
4	The semantics of “groups”	12
4.1	Semantic restrictions	13
5	Examples	14
5.1	Simple species typing via annotations	14
5.2	Example using meta identifiers	14
A	Validation of SBML documents using Groups constructs	17
	Acknowledgments	22
	References	23

1 Introduction

The SBML Level 3 Groups package offers a more flexible mechanism for indicating that components of an SBML model are related in some way. The nature of the relationship is left up to the modeler, and can be clarified by means of annotations in the model. Groups may contain either the same or different types of SBML objects, and groups may be nested if desired. There are no predefined behavioral semantics associated with groups.

1.1 Proposal corresponding to this package specification

This specification for Groups in SBML Level 3 Version 1 is based on the proposal located at the following URL:

<https://sbml.svn.sf.net/svnroot/sbml/trunk/specifications/sbml-level-3/version-1/groups/proposal>

The tracking number in the SBML issue tracking system (SBML Team, 2010) for Groups package activities is 2847474. The version of the proposal used as the starting point for this specification is the version of June, 2012 (Hucka, 2012).

1.2 Package dependencies

The Groups package has no dependencies on other SBML Level 3 packages.

1.3 Document conventions

UML 1.0 (Unified Modeling Language; Eriksson and Penker 1998; Oestereich 1999) notation is used in this document to define the constructs provided by this package. Colors in the diagrams carry the following additional information for the benefit of those viewing the document on media that can display color:

- **Black:** Items colored black in the UML diagrams are components taken unchanged from their definition in the SBML Level 3 Core specification document.
- **Green:** Items colored green are components that exist in SBML Level 3 Core, but are extended by this package. Class boxes are also drawn with dashed lines to further distinguish them.
- **Blue:** Items colored blue are new components introduced in this package specification. They have no equivalent in the SBML Level 3 Core specification.

The following typographical conventions distinguish the names of objects and data types from other entities; these conventions are identical to the conventions used in the SBML Level 3 Core specification document:

AbstractClass: Abstract classes are never instantiated directly, but rather serve as parents of other classes. Their names begin with a capital letter and they are printed in a slanted, bold, sans-serif typeface. In electronic document formats, the class names defined within this document are also hyperlinked to their definitions; clicking on these items will, given appropriate software, switch the view to the section in this document containing the definition of that class. (However, for classes that are unchanged from their definitions in SBML Level 3 Core, the class names are not hyperlinked because they are not defined within this document.)

Class: Names of ordinary (concrete) classes begin with a capital letter and are printed in an upright, bold, sans-serif typeface. In electronic document formats, the class names are also hyperlinked to their definitions in this specification document. (However, as in the previous case, class names are not hyperlinked if they are for classes that are unchanged from their definitions in the SBML Level 3 Core specification.)

Something, otherThing: Attributes of classes, data type names, literal XML, and tokens *other* than SBML class names, are printed in an upright typewriter typeface.

For other matters involving the use of UML and XML, this document follows the conventions used in the SBML Level 3 Core specification document.

2 Background and context

SBML Level 2 Versions 2–4 provides two object classes, **CompartmentType** and **SpeciesType**, meant to allow the definition of types of compartments and species. The original motivation for their introduction was the anticipation of the introduction of generalized reactions, a scheme that would have allowed reactions to be defined on whole classes of entities in a compact format. However, generalized reactions never ended up being introduced in SBML Level 2, and the notion of generalized reactions has been superceded by the effort to support rule-based models using the Level 3 Multistate and Multicomponent Species package. Moreover, the existence of just two types was never satisfactory in Level 2, because it did not satisfy the occasional desire to have other types of objects, such as parameters. For these reasons, when SBML Level 3 was developed, **CompartmentType** and **SpeciesType** were removed with the expectation that packages would be developed to take their place.

The SBML Level 3 Groups package is intended to fill the gaps left by the absence of **CompartmentType** and **SpeciesType** from SBML Level 3 Version 1 Core, while offering a more flexible mechanism for indicating that components of an SBML model are related. The nature of the relationship is left up to the modeler. This package can entirely take the place of **SpeciesType**, but the modeler must explicitly create the restrictions previously automatically associated with that element; as a trade-off, the Groups package offers modelers and software tools the ability to indicate relationships and restrictions between any type of SBML component, not just species and compartments.

The term *groups* is used in this package rather than *types*, because the latter would imply stronger behavioral constraints on objects than what the Groups package provides. This package really only provides a way of annotating a model. It does not provide a way to define types in the computer science sense; therefore, a different term is appropriate.

2.1 Prior work

The earliest relevant work on this topic is the development of the **CompartmentType** and **SpeciesType** object classes in the SBML Level 2 specification beginning with Version 2 (Finney et al., 2006). The original design was based on Andrew Finney's proposal for these object classes, which was made in the context of Finney's proposal for multicomponent species for SBML Level 3 (Finney, 2004). The **SpeciesType** class was included in the draft SBML Level 2 Version 2 to lay the groundwork for multicomponent species in Level 3, but **CompartmentType** was not; instead, a community vote was held in 2006 (SBML Editors, 2006a) on the question of whether **CompartmentType** should also be added to SBML Level 2 Version 2. The original expectation was to also introduce generalized reactions, and a community vote was held on this separate topic (SBML Editors, 2006b), but the result of the vote was that generalized reactions should be postponed to SBML Level 3 and not introduced in Level 2. Importantly, no one voted against generalized reactions outright, suggesting that a general typing mechanism for grouping species, compartments and reactions was something desired by the SBML community.

The first version of the Groups proposal was produced by the author in 2009 (Hucka, 2006), after discussions with the SBML Editors during the 2009 SBML Hackathon held at the European Bioinformatics Institute (Juty et al., 2009). The original proposal differed from the current proposal in several respects. The most notable architectural difference is that membership in groups was done in an inverted fashion compared to the current specification: model entities contained structures that indicated which groups they belonged to, rather than the current scheme, in which group definitions include lists of their members. The current scheme was developed during discussions with the SBML Editors during HARMONY 2011 in New York City (Anwar et al., 2011) and HARMONY 2012 in Maastricht, The Netherlands (Kutmon et al., 2012).

The idea for using an attribute to indicate the type of a group (i.e., a whether it is a classification, partonomy or simply a collection) was put forward by Nicolas Le Novère during the discussions at HARMONY 2012. The resulting attribute **kind** on the **Group** object class solved a problem of group meaning in the previous definitions of the Groups package.

Finally, the idea of adding constraints to a Group in order to fully recapture the original validation restrictions of **SpeciesType** was proposed by Lucian Smith, and incorporated into Release 0.2 of the specification.

3 Package syntax and semantics

This section contains a definition of the syntax and semantics of the Groups package for SBML Level 3 Version 1 Core. The Groups package involves six new object classes, **Group**, **Member**, **ListOfMemberConstraints**, **MemberConstraint**, **ListOfMembers** and **ListOfGroups**, as well as a simple extension of the existing **Model** object class. [Section 5 on page 14](#) contains complete examples of using the constructs in SBML models.

3.1 Namespace URI and other declarations necessary for using this package

Every SBML Level 3 package is identified uniquely by an XML namespace URI. For an SBML document to be able to use a given Level 3 package, it must declare the use of that package by referencing its URI. The following is the namespace URI for this version of the Groups package for SBML Level 3 Version 1 Core:

```
"http://www.sbml.org/sbml/level3/version1/groups/version1"
```

In addition, SBML documents using a given package must indicate whether the package can be used to change the mathematical interpretation of a model. This is done using the attribute **required** on the `<sbml>` element in the SBML document. For the Groups package, the value of this attribute must be **"false"**, because the use of the Groups package cannot change the mathematical meaning of a model.

The following fragment illustrates the beginning of a typical SBML model using SBML Level 3 Version 1 Core and this version of the Groups package:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
      xmlns:groups="http://www.sbml.org/sbml/level3/version1/groups/version1"
      groups:required="false">
```

3.2 Primitive data types

The Groups package uses a number of the primitive data types described in Section 3.1 of the SBML Level 3 Version 1 Core specification, and adds one additional primitive type described below.

3.2.1 Type GroupKind

The **GroupKind** primitive data type is used in the definition of the **Group** class. **GroupKind** is derived from type **string** and its values are restricted to being one of the following possibilities: **"classification"**, **"partonomy"**, and **"collection"**. Attributes of type **GroupKind** cannot take on any other values. The meaning of these three values is discussed in the context of the **Group** class' definition in [Section 3.3](#).

3.3 The Group class

The first and most central class in the Groups package is the **Group** class. [Figure 1 on the following page](#) provides the UML diagram of its definition. The **Group** class provides an optional identifier and name, one required attribute (**kind**), and two children: a list of members of the group, and a **ListOfMemberConstraints** child which imposes additional restrictions on those members. These are described below.

Since **Group** is derived from **SBase**, and **SBase** provides the ability to attach SBO terms as well as MIRIAM annotations, the semantics of a given group in a model can be made more precise by reference to external controlled vocabularies and ontologies. This capability is discussed further in [Section 4 on page 12](#).

3.3.1 The **id** and **name** attributes

The optional **id** attribute on the **Group** object class serves to provide a way to identify a group. The attribute takes a value of type **SId**. Note that the identifier of a group carries no mathematical interpretation and cannot be used in mathematical formulas in a model. **Group** also has an optional **name** attribute, of type **string**. The **name** attribute may be used in the same manner as other **name** attributes on SBML Level 3 Version 1 Core objects; please see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

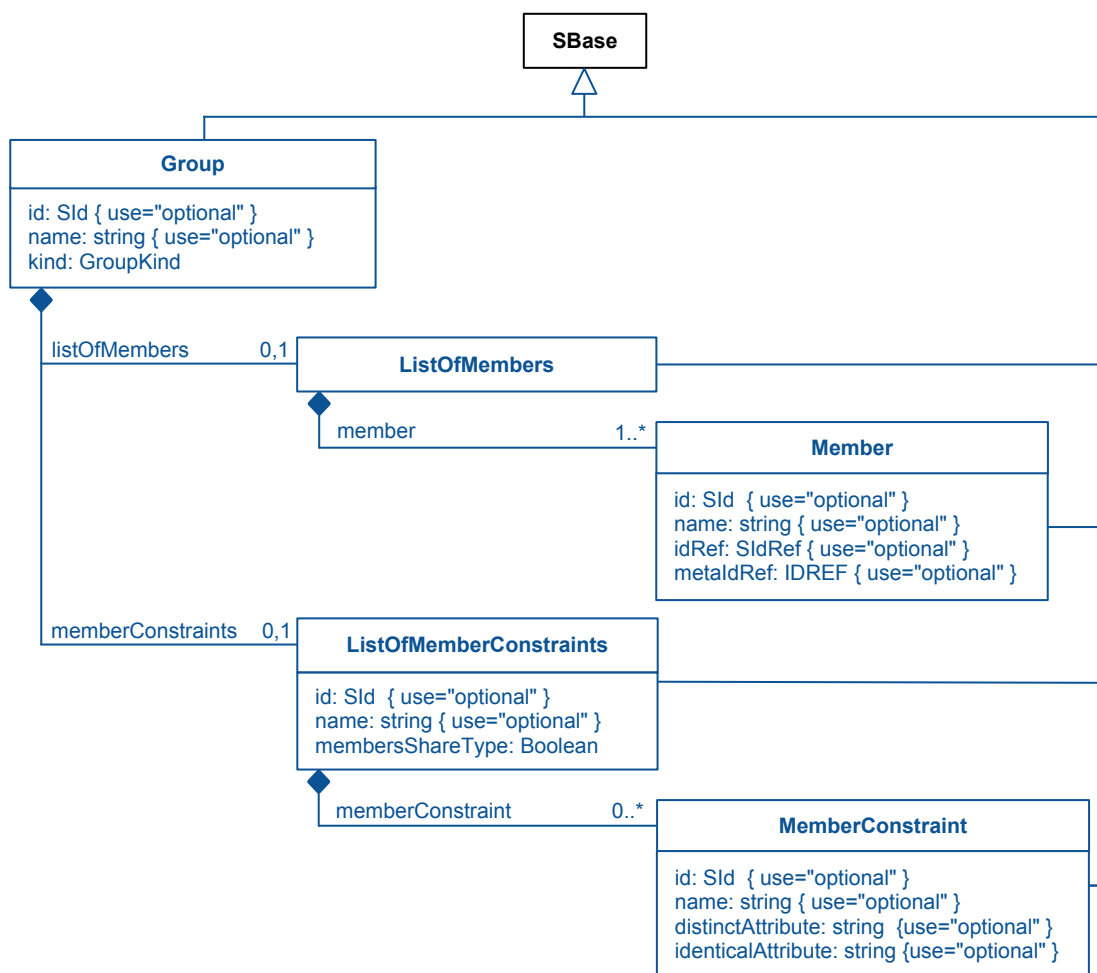


Figure 1: The definition of the **Group** class. The **ListOfMembers**, **Member**, **ListOfMemberConstraints**, and **MemberConstraint** classes are defined below.

3.3.2 The **kind** attribute

Group has one required attribute, **kind**, of type **GroupKind**. This attribute is used to indicate the nature of the group defined by a **Group** instance. The **kind** attribute must always have one of the three possible values of **GroupKind**; these values have the following meanings:

classification

The group represents a class, and its members have an *is-a* relationship to the group. For example, the group could represent a type of molecule such as ATP, and the members could be species located in different compartments, thereby establishing that the species are pools of the same molecule in different locations.

partonomy

The group represents a collection of parts, and its members have a *part-of* relationship to the group. For example, the group could represent a cellular structure, and individual compartments could be made members of the group to indicate they represent subparts of that cellular structure.

collection

The grouping is merely a collection for convenience, without an implied relationship between the members. For example, the group could be used to collect together multiple disparate components of a model—species, reactions, events—involved in a particular phenotype, and apply a common annotation rather than having to copy the same annotation to each component individually.

3.4 The **ListOfMembers** class

The **ListOfMembers** class is defined in Figure 1 on the preceding page, and must have one or more **Member** children. Since **ListOfMembers** is derived from **SBase**, it inherits the **sboTerm** and **metaId** attributes, as well as the optional children **Notes** and **Annotation**. Unlike most lists of objects in SBML, however, the **sboTerm** attribute and the **Notes** and **Annotation** children are taken here to apply directly to every SBML element referenced by each child **Member** of this **ListOfMembers**, if that referenced element has no such definition. If a referenced element has no defined **sboTerm**, that element should be considered to now have the **sboTerm** defined on the **ListOfMembers**. If a referenced element has no defined **Notes** child, that element should be considered to now have the **Notes** child of the **ListOfMembers**. However, since annotations can be cumulative, if a referenced element has no defined **Annotation** child, that element should be considered to now have the **Annotation** child of the **ListOfMembers**.

If multiple **ListOfMembers** reference the same SBML element, and more than one of them has a defined **sboTerm**, **Notes**, or **Annotation** element, those elements should be consistent with each other.

3.5 The **Member** class

The **Member** class is defined in Figure 1 on the previous page. It has two optional attributes, **id** and **name**, that allow others to reference it, and two optional attributes, **idRef** and **metaIdRef** that allow it to reference other elements, exactly *one* (and only one) of which must have a value in a given **Member** object instance. The referenced object (including, potentially, another **Group** object) is thus made a member of the group in which the **Member** object is contained. Multiple attributes are needed to account for the different types of identifiers that a given object may have. The meaning and purpose of the attributes on the object class are described below.

Since **Member** is derived from **SBase** and, as mentioned above, **SBase** provides both the ability to attach SBO terms as well as MIRIAM annotations, the semantics of a given member in a model can be made more precise by reference to external controlled vocabularies and ontologies.

3.5.1 The **id** and **name** attributes

The optional **id** attribute on the **Member** object class serves to provide a way to identify the member reference. The attribute takes a value of type **SIId**. Note that the identifier of a member reference carries no mathematical interpretation and cannot be used in mathematical formulas in a model. **Member** also has an optional **name** attribute, of type **string**. The **name** attribute may be used in the same manner as other **name** attributes on SBML Level 3 Version 1 Core objects; please see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

3.5.2 The **idRef** attribute

The attribute **idRef** on the **Member** class has type **SIIdRef**, and must be defined if the **Member** has no defined **metaIdRef** attribute, and must not be defined otherwise. The value must be the identifier of an object elsewhere in the **Model**. (Object identifiers are usually set by attributes named **id**; thus, the **idRef** value will usually be the **id** value of an object in the **Model**.) An example value of **idRef** might be the identifier of a species in the model, or the identifier of another group. The namespace in which the **SIId** is to be found is the **SIId** namespace

of the **Model** to which the **Group** belongs. This includes elements from SBML packages which may have elements with **id** values that are part of the **id** namespace of the **Model**, such as **Deletion** elements from the Hierarchical Model Composition package, **FluxBound** elements from the Flux Balance Constraints package, and **Group**, **Member**, **ListOfMemberConstraints**, and **MemberConstraint** elements from this Groups package. Conversely, elements with **id** values that are not part of the **id** namespace of the **Model** such as **Unit** and **LocalParameter** elements in SBML Level 3 Version 1 Core, or **Port** elements from the Hierarchical Model Composition package, may not be referenced by this **idRef** attribute.

3.5.3 The **metaIdRef** attribute

The **Member** attribute **metaIdRef** takes a value of type **IDREF**, and must be defined if the **Member** has no defined **idRef** attribute, and must not be defined otherwise. This attribute is used to refer to a **metaid** attribute value on any other object in the **Model**, for cases where the object being referenced does not have an identifier in the **Model** **id** namespace. (This is the case with, for example, rules in SBML Level 3 Version 1 Core.) Since meta identifiers are optional attributes of **SBase**, all SBML objects have the potential to have a meta identifier value, including most elements from other SBML packages.

Note that even if used in conjunction with the Hierarchical Model Composition package, this attribute is not allowed to reference elements from other **Model** objects in the same SBML Document. Referenced elements must be normal members of the parent **Model**, and submodel elements may be accessed normally by creating replacements.

3.5.4 Members referencing other Groups and Members

If a **Member** references a **Group**, **ListOfMembers**, or some other **Member** element, it is that referencing element itself that is considered to be a member of the **Group**, not the corresponding referenced element (or elements). This is also true for elements from other packages that point to other elements, such as **SBaseRef** elements from the Hierarchical Model Composition package.

The implication of this is that all the rules in this specification that apply to members of a group (such as how the **kind** attribute functions, the application of **sboTerm** attributes on a **ListOfMembers**, and the **ListOfMemberConstraints** restrictions, described below) apply to the child group itself, and not to its members. If a parent group includes two species plus a child group which itself contains three other species, if the parent group's **ListOfMembers** is given an **sboTerm**, that term applies to the two species and the group, *not* to the three child species members of the second group. That parent group's **kind** would also almost certainly be “**collection**” or “**partonomy**”, and not “**classification**”, as two species and a group are very unlikely to be classified as the same thing.

In a future version of this specification, it may be possible to perform set operations on groups. At that point, it will be possible to create set unions, which might then be further manipulated, analyzed, or annotated.

Finally, groups are not permitted to be circular: no **Member** may reference itself, its parent **ListOfMembers**, nor its parent **Group**. If a **Member** references a **Group**, the same restrictions apply to that subgroup's children: they may not reference the **Member**, its parent **ListOfMembers**, nor its parent **Group**, and if any of those children reference a **Group**, the same restrictions apply to them, etc.

3.5.5 Members referencing other namespaces

If a **Member** has a **idRef** or **metaIdRef** attribute which references an object from a namespace that is not understood by the interpreter, that **Member** must be ignored—the referenced object will not be understood by the interpreter, and therefore has no need to become a member of the group. If an interpreter cannot tell whether a referenced object does not exist or if exists in an unparsed namespace, it may choose to produce a warning.

3.5.6 Example

As mentioned above, exactly one of the attributes **idRef** and **metaIdRef** must have a value in a given **Member** object instance. There are no restrictions on mixing the attributes used across multiple members of the same group, however. The following artificial example illustrates the use of heterogeneous and nested groups:


```

...
<listOfSpecies>
  <species id="S1" compartment="c" initialConcentration="1"/>
  <species id="S2" compartment="c" initialConcentration="2"/>
</listOfSpecies>
<listOfCompartments>
  <compartment id="C" size="1"/>
</listOfCompartments>
<listOfRules>
  <rateRule variable="S2" metaid="_rule1">
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply>
        <times/>
        <ci> S1 </ci>
        <cn> 0.5 </cn>
      </apply>
    </math>
  </rateRule>
</listOfRules>
<listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
  <group id="all_species" kind="collection">
    <listOfMembers>
      <member idRef="S1"/>
      <member idRef="S2"/>
    </listOfMembers>
  </group>
  <group id="all_entities" kind="collection">
    <listOfMembers>
      <member idRef="all_species"/>
      <member idRef="C"/>
      <member metaIdRef="_rule1"/>
    </listOfMembers>
  </group>
</listOfGroups>
...

```

3.6 The `ListOfMemberConstraints` class

The `ListOfMemberConstraints` class is defined in Figure 1 on page 6 as a way to impose restrictions on the referenced member elements of a `Group`. It can be used to ensure that all members of a group are of the same SBML Class or to ensure that all members of the group have the same attribute with the same value or with all different values.

It has one required attribute, `membersShareType`, and two optional attributes: `id` and `name`. It also may contain one or more `MemberConstraint` child objects, all described below.

3.6.1 The `id` and `name` attributes

The optional `id` attribute on the `ListOfMemberConstraints` object class serves to provide a way to identify the constraints. The attribute takes a value of type `SIId`. Note that this identifier carries no mathematical interpretation and cannot be used in mathematical formulas in a model. `ListOfMemberConstraints` also has an optional `name` attribute, of type `string`. The `name` attribute may be used in the same manner as other `name` attributes on SBML Level 3 Version 1 Core objects; please see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

3.6.2 The `membersShareType` attribute

The required attribute `membersShareType` on the `ListOfMemberConstraints` class has type `boolean`. If set to “true”, all referenced members of the `Group` must be the same exact SBML class. This means that sharing a base class (such as `SBase` or `Rule`) is insufficient: an `AlgebraicRule` and a `RateRule` are different classes from one another.

If set to “false”, the referenced members of the **Group** may be of any SBML class; no restrictions are implied.

This restriction can only be violated if there are two or more members of the group.

3.7 The **MemberConstraint** class

The **MemberConstraint** class is defined in Figure 1 on page 6 as a way to impose restrictions on common attributes of the referenced members of the **Group**. It has two optional attributes to identify the attribute constraint itself (**id** and **name**), and two other optional attributes (**identicalAttribute** and **distinctAttribute**) of which exactly *one* (and only one) must be defined.

3.7.1 The **id** and **name** attributes

The optional **id** attribute on the **MemberConstraint** object class serves to provide a way to identify the **MemberConstraint**. The attribute takes a value of type **SId**. Note that this identifier carries no mathematical interpretation and cannot be used in mathematical formulas in a model. **MemberConstraint** also has an optional **name** attribute, of type **string**. The **name** attribute may be used in the same manner as other **name** attributes on SBML Level 3 Version 1 Core objects; please see Section 3.3.2 of the SBML Level 3 Version 1 Core specification for more information.

3.7.2 The **distinctAttribute** attribute

The attribute **distinctAttribute** on the **MemberConstraint** class has type **string**. This attribute must be defined if the **MemberConstraint** has no defined **identicalAttribute**, and must not be defined otherwise. The value of the string must be the name of an attribute shared by all referenced members of the **Group**. Furthermore, the *values* of those attribute must be different across all referenced elements. The rule from SBML Level 2 that no two **SpeciesType** elements may share a compartment could be replicated here by setting this attribute to “**compartment**”.

If the referenced attribute exists in a particular namespace, the namespace should be included as part of the string, with a colon separating it from the text of the attribute name itself (i.e. “**groups:idRef**”).

This restriction can only be violated if there are two or more members of the group.

3.7.3 The **identicalAttribute** attribute

The attribute **identicalAttribute** on the **MemberConstraint** class has type **string**. This attribute must be defined if the **MemberConstraint** has no defined **distinctAttribute**, and must not be defined otherwise. The value of the string must be the name of an attribute shared by all referenced members of the **Group**. Furthermore, the *values* of those attributes must be the same for all referenced elements. If the referenced members are all **Species**, for example, this attribute might be set to be “**boundaryCondition**”, ensuring that all the referenced elements have the same value (“**true**” or “**false**”). To ensure that all of these species had the same units, another **MemberConstraint** could be set with an **identicalAttribute** of “**substanceUnits**”.

If the referenced attribute exists in a particular namespace, the namespace should be included as part of the string, with a colon separating it from the text of the attribute name itself (i.e. “**groups:idRef**”).

This restriction can only be violated if there are two or more members of the group.

3.7.4 Example

A partial example of a working **ListOfMemberConstraints** element is shown below. In it, the two referenced members (“**ATPc**” and “**ATPm**”) are declared to be the same type using the **membersShareType** attribute, that they must have different compartments with a **distinctAttribute** attribute, and that they must both have the same units with a **identicalAttribute** attribute.

```

<group id="ATP" kind="classification">
  <listOfMembers sboTerm="SB0:0000248">
    <member idRef="ATPc" />
    <member idRef="ATPm" />
  </listOfMembers>
  <listOfMemberConstraints membersShareType="true">
    <memberConstraint distinctAttribute="compartment" />
    <memberConstraint identicalAttribute="substanceUnits" />
  </listOfMemberConstraints>
</group>

```

A complete SBML document including the above snippet is included in [Section 5.1 on page 14](#).

3.8 The extended **Model** class

The Groups package extends SBML Level 3 Version 1 Core's **Model** class to add one list, **ListOfGroups**, for holding group definitions. [Figure 2](#) provides the UML diagram for the extension.

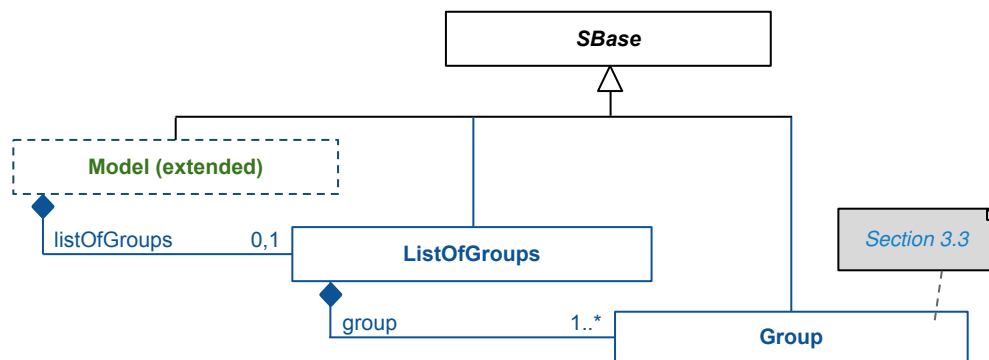


Figure 2: The extensions of the **Model** class. **Group** is defined in [Section 3.3 on page 5](#). In other respects, **Model** remains defined as in the SBML Level 3 Version 1 Core specification.

3.8.1 The list of groups

[Figure 2](#) shows that the extension of **Model** by the Groups package involves adding optional **listOfGroups** subcomponent for holding a **ListOfGroups** container object. If present, the **ListOfGroups** instance must contain at least one **Group** object ([Section 3.3 on page 5](#)). In common with other **ListOf__** classes in SBML, **ListOfGroups** is derived from **SBase**. It inherits **SBase**'s attributes **metaid** and **sboTerm**, as well as the subcomponents for **Annotation** and **Notes**, but does not add any new attributes of its own.

4 The semantics of “groups”

A group *G* is defined by declaring an instance of a **Group** class object within the **ListOfGroups** element of a **Model** object. The group can be given an optional identifier (i.e., a value for its **id** attribute), but even if the group does not have an identifier, the act of declaring a group has the effect of creating it.

An entity *X* in the model is declared to be part of group *G* by listing the identifier of *X* in a **Member** object within the **ListOfGroups** instance of *G*. The following is an example to illustrate the structure:

```
<model id="model_1">
  <listOfSpecies>
    <species id="s1" .../>
    <species id="s2" .../>
    <species id="s3" .../>
    <species id="s4" .../>
  </listOfSpecies>
  ...
  <listOfReactions>
    <reaction id="r1" ...> ... </reaction>
    <reaction id="r2" ...> ... </reaction>
  </listOfReactions>
  ...
  <listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
    <group id="some_species_group" kind="collection">
      <listOfMembers>
        <member idRef="s1"/>
        <member idRef="s3"/>
      </listOfMembers>
    </group>
    <group id="some_reaction_group" kind="collection">
      <listOfMembers>
        <member idRef="r1"/>
        <member idRef="r2"/>
      </listOfMembers>
    </group>
  </listOfGroups>
</model>
```

For any given group, the meaning of group membership is determined by the value of the attribute **kind** on the **Group** object instance. Examples of possible meanings are given in [Section 3.3.2 on page 6](#).

The meaning of a group can be further refined by using annotations (either SBO terms or the **Annotation** element) on the group, or the list of members. The following are the interpretations. This possibility raises the question of how the annotations should be interpreted across the group members. The following are the possibilities defined by the Groups package:

- If the annotation or SBO term is on a **Group** object, it is an annotation about the group itself, not the individual members.
- If the annotation or SBO term is on a **Member** object, it is an annotation specifically about that member, and not about any other member nor the group overall.
- If the annotation or SBO term is on **ListOfMembers**, it is a short-hand that means the annotation applies to each individual member, as if the annotation were put on the individual members directly.

Finally, as mentioned in [Section 3.5.4 on page 8](#), groups can refer to other groups, leading to the possibility of group hierarchies. To indicate a group is part of another group, one simply needs to include the first group's identifier as a member of the second group, as in the following example:

```

<model id="model_2">
  ...
  <listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
    <group id="group1">
      <listOfMembers>
        <member idRef="...">
        <member idRef="...">
      </listOfMembers>
    </group>
    <group id="group2">
      <listOfMembers>
        <member idRef="group1"/>
        <member idRef="...">
        <member idRef="...">
      </listOfMembers>
    </group>
  </listOfGroups>
</model>

```

The intended meaning of nested groups can be made more precise by annotating the group and list of members with appropriate MIRIAM annotations using controlled vocabulary terms that describe the meaning.

4.1 Semantic restrictions

The current definition of the Groups package is such that the use of Groups constructs has no impact on the mathematics of a model. As a consequence of this, this package is not required for proper interpretation of a model. Models must use the **required="false"** flag on the declaration of the package on the **<sbml>** element in a file.

Because restrictions can be imposed on the group through the use of the **ListOfMemberConstraints** element, the Groups package can fully recapture SBML Level 2's **SpeciesType** construct: to do so, the group's **kind** attribute should be set to "classification", a **ListOfMemberConstraints** should be added with its **membersShareType** attribute set to "true", and it should be given a child **MemberConstraint** with its **distinctAttribute** attribute set to "compartment". A fully worked example (which also imposes additional restrictions) is given in [Section 5.1 on the following page](#).

5 Examples

This section contains examples employing the Groups package for SBML Level 3.

5.1 Simple species typing via annotations

The following is a simple example of using this proposed grouping facility to do something similar to the **SpeciesType** example shown in Section 4.6.3 of the SBML Level 2 Version 4 specification (p. 43).

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
      xmlns:groups="http://www.sbml.org/sbml/level3/version1/groups/version1"
      groups:required="false">
  <model>
    <listOfSpecies>
      <species id="ATPc" compartment="cytosol" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true" />
      <species id="ATPm" compartment="mitochon" substanceUnits="mole" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true" />
    </listOfSpecies>
    <listOfCompartments>
      <compartment id="cytosol" spatialDimensions="3" size="1" constant="true" />
      <compartment id="mitochon" spatialDimensions="3" size="1" constant="true" />
    </listOfCompartments>
    <listOfGroups xmlns="http://www.sbml.org/sbml/level3/version1/groups/version1">
      <group id="ATP" kind="classification">
        <listOfMembers sboTerm="SBO:0000248">
          <member idRef="ATPc" />
          <member idRef="ATPm" />
        </listOfMembers>
        <listOfMemberConstraints membersShareType="true">
          <memberConstraint distinctAttribute="compartment" />
          <memberConstraint identicalAttribute="initialConcentration" />
          <memberConstraint identicalAttribute="constant" />
        </listOfMemberConstraints>
      </group>
    </listOfGroups>
  </model>
</sbml>
```

In this example, both species “ATPc” and “ATPm” are intended to be pools of ATP, but located in different compartments. To indicate that they are both conceptually the same kind of molecular entity, the model includes a group definition of the “classification” variety. The two species “ATPc” and “ATPm” are both listed as members of the same group. The **ListOfMembers** is given the **sboTerm** “SBO:0000248” to indicate that both species are small molecules. Four restrictions are set in place by the **ListOfMemberConstraints** element: both species must be the same type, must be in different **compartments**, and must share the same **initialConcentration** and **constant** values.

The group definition could be enhanced further by including an annotation on the **ListOfMembers** that references the ChEBI database entry for ATP; we omit that detail here in order to concentrate on the Groups constructs.

5.2 Example using meta identifiers

In the next example, a group is used to annotate two rules to annotate the fact that the two rules influence a model for some particular reason.

```

<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level3/version1/core" level="3" version="1"
  xmlns:groups="http://www.sbml.org/sbml/level3/version1/groups/version1"
  groups:required="false">
  <model id="cell" name="cell">
    <listOfCompartments>
      <compartment id="compartment" spatialDimensions="3" size="1" constant="true" />
    </listOfCompartments>
    <listOfSpecies>
      <species id="Dose1" compartment="compartment" initialConcentration="0" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true" />
      <species id="Dose2" compartment="compartment" initialConcentration="0" constant="false"
        hasOnlySubstanceUnits="false" boundaryCondition="true"/>
    </listOfSpecies>
    <listOfRules>
      <assignmentRule metaid="D1" variable="Dose1">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <piecewise>
            <piece>
              <cn type="integer"> 2 </cn>
              <apply>
                <lt />
                <csymbol encoding="text"
                  definitionURL="http://www.sbml.org/sbml/symbols/time"> time </csymbol>
                <cn type="integer"> 1 </cn>
              </apply>
            </piece>
            <otherwise>
              <cn type="integer"> 0 </cn>
            </otherwise>
          </piecewise>
        </math>
      </assignmentRule>
      <assignmentRule metaid="D2" variable="Dose2">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <piecewise>
            <piece>
              <cn> 1.5 </cn>
              <apply>
                <and />
                <apply>
                  <gt />
                  <csymbol encoding="text"
                    definitionURL="http://www.sbml.org/sbml/symbols/time"> time </csymbol>
                  <cn type="integer"> 5 </cn>
                </apply>
              </apply>
            </piece>
            <otherwise>
              <cn type="integer"> 0 </cn>
            </otherwise>
          </piecewise>
        </math>
      </assignmentRule>
    </listOfRules>
    <groups:listOfGroups>
      <groups:group groups:id="effectB" groups:kind="collection">
        <notes>
          <p xmlns="xhtmlnamespace"> These two rules are in the model as approximation
            for effect B</p>
        </notes>
      </groups:group>
    </groups:listOfGroups>
  </model>
</sbml>

```



```
</notes>
<groups:listOfMembers>
  <groups:member groups:metaIdRef="D1"/>
  <groups:member groups:metaIdRef="D2"/>
</groups:listOfMembers>
</groups:group>
</groups:listOfGroups>
</model>
</sbml>
```

The key point of this example is the use of meta identifiers for SBML entities (in particular, rules) that do not have regular identifiers (i.e., `id` attributes).

A Validation of SBML documents using Groups constructs

This section summarizes all the conditions that must (or in some cases, at least *should*) be true of an SBML Level 3 Version 1 model that uses the Groups package. We use the same conventions that are used in the SBML Level 3 Version 1 Core specification document. In particular, there are different degrees of rule strictness. Formally, the differences are expressed in the statement of a rule: either a rule states that a condition *must* be true, or a rule states that it *should* be true. Rules of the former kind are strict SBML validation rules—a model encoded in SBML must conform to all of them in order to be considered valid. Rules of the latter kind are consistency rules. To help highlight these differences, we use the following three symbols next to the rule numbers:

- ☑ A checked box indicates a *requirement* for SBML conformance. If a model does not follow this rule, it does not conform to the Groups package specification. (Mnemonic intention behind the choice of symbol: “This must be checked.”)
- ▲ A triangle indicates a *recommendation* for model consistency. If a model does not follow this rule, it is not considered strictly invalid as far as the Groups package specification is concerned; however, it indicates that the model contains a physical or conceptual inconsistency. (Mnemonic intention behind the choice of symbol: “This is a cause for warning.”)
- ★ A star indicates a strong recommendation for good modeling practice. This rule is not strictly a matter of SBML encoding, but the recommendation comes from logical reasoning. As in the previous case, if a model does not follow this rule, it is not considered an invalid SBML encoding. (Mnemonic intention behind the choice of symbol: “You’re a star if you heed this.”)

The validation rules listed in the following subsections are all stated or implied in the rest of this specification document. They are enumerated here for convenience. Unless explicitly stated, all validation rules concern objects and attributes specifically defined in the Groups package.

- ✎ For convenience and brevity, we use the shorthand “**groups:x**” to stand for an attribute or element name **x** in the namespace for the Groups package, using the namespace prefix **groups**. In reality, the prefix string may be different from the literal “**groups**” used here (and indeed, it can be any valid XML namespace prefix that the modeler or software chooses). We use “**groups:x**” because it is shorter than to write a full explanation everywhere we refer to an attribute or element in the Groups package namespace.

General rules about the Groups package

- groups-10101** ☑ To conform to Version 1 of the Groups package specification for SBML Level 3, an SBML document must declare the use of the following XML Namespace: “<http://www.sbml.org/sbml/level3/version1/groups/version1>”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.1 on page 5](#).)
- groups-10102** ☑ Wherever they appear in an SBML document, elements and attributes from the Groups package must be declared either implicitly or explicitly to be in the XML namespace “<http://www.sbml.org/sbml/level3/version1/groups/version1>”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.1 on page 5](#).)

Rules for the extended SBML class

- groups-10201** ☑ In all SBML documents using the Groups package, the **SBML** object must include a value for the attribute **groups:required** attribute. (References: SBML Level 3 Version 1 Core, Section 4.1.2.)
- groups-10202** ☑ The value of attribute **groups:required** on the **SBML** object must be of the data type **boolean**. (References: SBML Level 3 Version 1 Core, Section 4.1.2.)

- groups-10203** ✓ The value of attribute **groups:required** on the **SBML** object must be set to “false” (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.1 on page 5.](#))

General rules about attributes

- groups-10301** ✓ (Extends validation rule #10301 in the SBML Level 3 Version 1 Core specification.) Within a **Model** object, the values of the attributes **id** and **groups:id** on every instance of the following classes of objects must be unique across the set of all **id** and **groups:id** attribute values of all such objects in a model: the **Model** itself, plus all contained **FunctionDefinition**, **Compartment**, **Species**, **Reaction**, **SpeciesReference**, **ModifierSpeciesReference**, **Event**, and **Parameter** objects, plus the **Group**, **Member**, **ListOfMemberConstraints**, and **MemberConstraint** objects defined by the Groups package, plus any objects defined by any other package with **package:id** attributes defined as falling in the ‘Sid’ namespace. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-10302** ✓ The value of a **groups:id** attribute must always conform to the syntax of the SBML data type **SId**. (References: SBML Level 3 Version 1 Core, Section 3.1.7.)
- groups-10303** ✓ The value of a **groups:name** attribute must always conform to the syntax of type **string**. (References: SBML Level 3 Version 1 Core, Section 3.1.1.)
- groups-10304** ✓ The value of a **groups:idRef** attribute on **Member** objects must conform to the syntax of the SBML data type **SIdRef**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.5 on page 7.](#))
- groups-10305** ✓ The value of a **groups:metaIdRef** attribute on **Member** objects must conform to the syntax of the SBML data type **IDREF**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.5 on page 7.](#))
- groups-10306** ✓ The value of a **groups:membersShareType** attribute on **ListOfMemberConstraints** objects must be of the data type **boolean**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.6 on page 9.](#))
- groups-10307** ✓ The value of a **groups:identicalAttribute** attribute on **MemberConstraint** objects must be of the data type **string**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.6 on page 9.](#))
- groups-10308** ✓ The value of a **groups:distinctAttribute** attribute on **MemberConstraint** objects must be of the data type **string**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.6 on page 9.](#))

Rules for extended Model objects

- groups-20101** ✓ There may be at most one **ListOfGroups** container object within a **Model** object, named “listOfGroups”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.8 on page 11.](#))
- groups-20102** ✓ A **ListOfGroups** object within a **Model** object is optional, but if present, must not be empty. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.8 on page 11.](#))
- groups-20103** ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfGroups** container object may only contain **Group** objects, named “group”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.8 on page 11.](#))
- groups-20104** ✓ A **ListOfGroups** object may have the optional SBML core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespace or the Groups namespace are

permitted on a **ListOfGroups** object. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.8 on page 11.](#))

Rules for Group objects

- groups-20201** ✓ A **Group** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespace are permitted on a **Group** object. (References: SBML Level 3 Version 1 Core, [Section 3.2.](#))
- groups-20202** ✓ A **Group** object may have the attributes **groups:id** and **groups:name**, and must have the attribute **groups:kind**. No other attributes from the Groups namespace are permitted on a **Group** object. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-20203** ✓ The value of the **groups:kind** attribute on a **Group** object must have one of the following values: “**classification**”, “**partonomy**”, or “**collection**”. These are the only permitted values for the **groups:kind** attribute on **Group**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-20204** ✓ There may be at most one **ListOfMembers** container object within a **Group** object, named “**listOfMembers**”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-20205** ✓ A **ListOfMembers** object within a **Group** object is optional, but if present, must not be empty. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-20206** ✓ Apart from the general notes and annotation subobjects permitted on all SBML objects, a **ListOfMembers** container object may only contain **Member** objects, named “**member**”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-20207** ✓ A **ListOfMembers** object may have the optional SBML core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespace or the Groups namespace are permitted on a **ListOfMembers** object. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-20208** ✓ There may be at most one **ListOfMemberConstraints** object within a **Group** object, with the name “**listOfMemberConstraints**”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.3 on page 5.](#))
- groups-20209** ▲ If multiple **ListOfMembers** objects contain **Member** elements that reference the same SBML object, the **sboTerm** and any child **Notes** or **Annotation** elements set for those **ListOfMembers** should be consistent, as they all should apply to the same referenced object. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.4 on page 7.](#))

Rules for Member objects

- groups-20301** ✓ A **Member** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespace are permitted on a **Group** object. (References: SBML Level 3 Version 1 Core, [Section 3.2.](#))
- groups-20302** ✓ A **Member** object must have a value for one (and exactly one) of the attributes **groups:idRef** or **groups:metaIdRef**, and may additionally have the attributes **groups:id** and **groups:name**. No other attributes from the Groups namespace are permitted on a **Member** object. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.5 on page 7.](#))
- groups-20303** ✓ The value of the **groups:idRef** attribute, if set on a given **Member** object, must be the value of

an **id** attribute on an existing object in the **SI**d namespace of the parent **Model**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.5 on page 7.](#))

- groups-20304** ✓ The value of the **groups:metaIdRef** attribute, if set on a given **Member** object, must be the value of a meta identifier on an existing object in the parent **Model**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.5 on page 7.](#))

Rules for ListOfMemberConstraints objects

- groups-20401** ✓ A **ListOfMemberConstraints** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespace are permitted on a **Group** object. (References: SBML Level 3 Version 1 Core, Section 3.2.)
- groups-20402** ✓ A **ListOfMemberConstraints** object must have the attribute **groups:membersShareType**, and may additionally have the attributes **groups:id** and **groups:name**. No other attributes from the Groups namespace are permitted on a **ListOfMemberConstraints** object. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.6 on page 9.](#))
- groups-20403** ✓ A **ListOfMemberConstraints** object may contain one or more **MemberConstraint** children, named “**memberConstraint**”. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.6 on page 9.](#))

Rules for MemberConstraint objects

- groups-20501** ✓ A **MemberConstraint** object may have the optional SBML Level 3 Core attributes **metaid** and **sboTerm**. No other attributes from the SBML Level 3 Core namespace are permitted on a **Group** object. (References: SBML Level 3 Version 1 Core, Section 3.2.)
- groups-20502** ✓ A **MemberConstraint** object must have a value for one (and exactly one) of the attributes **groups:distinctAttribute** and **groups:identicalAttribute**, and may additionally have the attributes **groups:id** and **groups:name**. No other attributes from the Groups namespace are permitted on a **MemberConstraint** object. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.7 on page 10.](#))
- groups-20503** ✓ The value of the **groups:distinctAttribute** attribute, if set on a given **MemberConstraint** object, must be the name of an attribute shared by all referenced members of its parent **Group**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.7.2 on page 10.](#))
- groups-20504** ✓ The value of the **groups:identicalAttribute** attribute, if set on a given **MemberConstraint** object, must be the name of an attribute shared by all referenced members of its parent **Group**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.7.3 on page 10.](#))
- groups-20505** ✓ The values of any attribute named by a **groups:distinctAttribute** attribute on an **MemberConstraint** object must be different across all referenced members of that **MemberConstraint**’s parent **Group**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.7.2 on page 10.](#))
- groups-20506** ✓ The values of any attribute named by a **groups:identicalAttribute** attribute on an **MemberConstraint** object must be the same across all referenced members of that **MemberConstraint**’s parent **Group**. (References: SBML Level 3 Package Specification for Groups, Version 1, [Section 3.7.3 on page 10.](#))

Acknowledgments

42

Work such as this does not take place in a vacuum; many people contributed ideas and discussions that shaped the Groups proposal that you see before you. We particularly thank Nicolas Le Novère, Frank Bergmann, Sarah Keating, Allyson Lister, Robert Phair, Sven Sahle, Stefan Hoops, Chris Myers, and the members of the *sbml-discuss* and *sbml-groups* mailing lists for suggestions and comments.

1

2

3

4

This work was funded by the National Institutes of Health (USA) under grant R01 GM070923.

5

References

- Anwar, N., Demir, E., Hucka, M., and Novère, N. L. (2011). HARMONY 2011: The HAcKathon on Resources for MOdeliNg in biologY. Available via the World Wide Web at http://www.co.mbine.org/events/HARMONY_2011.
- Eriksson, H.-E. and Penker, M. (1998). *UML Toolkit*. John Wiley & Sons, New York.
- Finney, A. (2004). Multicomponent Species: A proposal for SBML Level 3. Available via the World Wide Web at <http://sbml.org/images/1/19/20041015-finney-multicomponent.pdf>.
- Finney, A., Hucka, M., and Novère, N. L. (2006). The Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. Available via the World Wide Web at <http://sbml.org/Documents/Specifications>.
- Hucka, M. (2006). Groups proposal (2009-09). Available via the World Wide Web at http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Groups_Proposal_%282009-09%29.
- Hucka, M. (2012). Groups proposal updated (2012-06). Available via the World Wide Web at http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Groups_Proposal_Updated_%282012-06%29.
- Juty, N., Novère, N. L., Taddeo, L., and Hucka, M. (2009). Seventh SBML Hackathon. Available via the World Wide Web at http://sbml.org/Events/Hackathons/The_7th_SBML_Hackathon.
- Kutmon, M., Evelo, C., Bader, G., Novère, N. L., and Hucka, M. (2012). HARMONY 2012: The HAcKathon on Resources for MOdeliNg in biologY. Available via the World Wide Web at http://www.co.mbine.org/events/HARMONY_2012.
- Oestereich, B. (1999). *Developing Software with UML: Object-Oriented Analysis and Design in Practice*. Addison-Wesley.
- SBML Editors, T. (2006a). Results of L2v2 specification vote #8: Introducing CompartmentType. Available via the World Wide Web at <http://sbml.org/Forums/index.php?t=tree&goto=3057&rid=2>.
- SBML Editors, T. (2006b). Results of L2v2 specification vote #9: Introducing Generalized Reactions. Available via the World Wide Web at <http://sbml.org/Forums/index.php?t=tree&th=764&mid=5567&rid=2>.
- SBML Team (2010). The SBML issue tracker. Available via the World Wide Web at <http://sbml.org/issue-tracker>.