# A short description of the main differences between JSBML and LibSBML

Andreas Dräger

November 5, 2010

Although JSBML and LibSBML are very similar, this document gives a brief overview of the main differences between the Java$^{TM}$ application programming interfaces of both libraries.

## 1 History

In earlier versions of SBML only the model itself could be associated with a history, i.e., a description about the person(s) who build this model, including names, e-mail addresses, modification and creation dates. Nowadays, it has become possible to annotate each individual construct of an SBML model with such a history. This is reflected by naming the corresponding object `History` in JSBML, whereas it is still called `ModelHistory` in LibSBML. Hence, all instances of `SBase` in JSBML contain methods do access and manipulate its `History`.

## 2 Deprecation

The intension of JSBML is to provide a Java library for the latest specification of SBML. Hence, JSBML provides methods and classes to cover earlier releases of SBML as well, but these are often marked as being deprecated to avoid creating models that refer to these elements.

## 3 UnitDefinitions

A model in JSBML always also contains all predefined units in the model if there are any, i.e., for models encoded of SBML versions before level 3. These can be accessed from an instance of model by calling the method `getPredefinedUnit(String unit)`.

## 4 MathContainer

This interface gathers all those elements that may contain mathematical expressions encoded in abstract syntax trees (instances of `ASTNode`). The abstract class `AbstractMathContainer` serves as actual super class for most of the derived types.

## 5 ASTNodeCompiler

This interface allows users to create customized interpreters for the content of mathematical equations encoded in abstract syntax trees. It is directly and recursively called from the `ASTNode` class and returns an `ASTNodeValue` object, which wraps the possible evaluation results of the interpretation. JSBML already provides several implementations of this interface, for instance, `ASTNode` objects can be directly translated to LaTeX or MathML for further processing.

## 6 InitialAssignment

JSBML unifies all those elements that assign values to some other `SBase` in SBML under the interface Assignment. This interface uses the term Variable for the element whose value is to be changed depending on some mathematical expression that is also present in the Assignment (because Assignment extends the interface MathContainer). Therefore, an Assignment contains methods such as `set/getVariable(Variable v)` and also `isSetVariable()` and `unsetVariable()`. In addition to that JSBML also provides the method `set/getSymbol(String symbol)` in the InitialAssignment class to make sure that switching from LibSBML to JSBML is quite smoothly. However, the preferred way in JSBML is to apply the methods setVariable either with String or Variable instances as arguments.

## 7 The class LibSBML

There is no class LibSBML because this library is called JSBML. You can therefore only find a class JSBML. This class provides similar methods as the LibSBML class in LibSBML.

## 8 LibSBMLConstants

You won't find a corresponding implementation of this interface in JSBML. The reason is that the JSBML team decided to encode constants using the Java construct enum. For instance, all the fields starting with the prefix `AST_TYPE_*` have a corresponding field in the `ASTNode` class itself. There you can find the Type enum. Instead of typing `AST_TYPE_PLUS`, you would therefore type `ASTNode.Type.PLUS`.

The same holds true for `Unit.Kind.*` corresponding to the `LibSBMLConstants.UNIT_KIND_*` fields.

## 9 ListOf

There is no method `get(String id)` because the generic implementation of the `ListOf<? extends SBase>` class in JSBML excepts also elements that do not necessarily have an identifier. Only instances of `NamedSBase` may have the fields identifier and name set. Hence, generally, the `ListOf` class cannot assume these fields to be present. To query an instance of `ListOf` in JSBML for names or identifiers or both, you can apply the following filter:

```
NamedSBase nsb = myList.firstHit(new NameFilter(identifier));
```

This will give you the first element in the list with the given identifier. Various filters are already implemented, but you can easily add your customized filter. To this end, you only have to implement the `Filter` interface in `org.sbml.jsbml.util.filters`. There you can also find an `OrFilter` and an `AndFilter`, which take as arguments multiple other filters. With the `SBOFilter` you can query for certain SBO annotations in your list, whereas the `CVTermFilter` helps you to identify `SBase` instances with a desired MIRIAM annotation. For instances of `ListOf<Species>` you can apply the `BoundaryConditionFilter` to look for those species that operate on the boundary of the reaction system.