SBtab table format – Version 1.2

Wolfram Liebermeister[1,2] and Timo Lubitz

[1] Université Paris-Saclay, INRAE, MaIAGE, 78350, Jouy-en-Josas, France
[2] Institut für Biochemie, Charité - Universitätsmedizin Berlin

Abstract

Data tables in the form of spreadsheets or delimited text files are the most utilised data format in Systems Biology. However, such tables are often not sufficiently structured and lack clear naming conventions that would be required for modelling. We propose the SBtab format as an attempt to establish an easy-to-use table format that is both flexible and clearly structured. It comprises defined table types for different kinds of data; syntax rules for usage of names, element IDs, and database identifiers used for annotation; and standardised formulae for reaction stoichiometries. Predefined table types can be used to define biochemical network models and the biochemical constants therein. The user can also define own table types, adjusting SBtab to other types of data. Software code, tools, and further information can be obtained at www.sbtab.net.

Draft version – March 2020

For updates and further information, please visit www.sbtab.net.

Contents

1 Introduction **1**

2 General form of the SBtab format **4**

2.1 SBtab format and biological data types . . . 4

2.2 SBtab documents and SBtab tables . . . 4

2.3 Use of comment lines . . . 6

2.4 Syntax rules for mathematical expressions . . . 6

2.5 File formats . . . 6

2.6 Defining an SBtab format . . . 7

2.7 MIRIAM-compliant models . . . 8

3 SBtab formats for biological models and data **9**

3.1 Biochemical networks and models . . . 9

3.2 Biochemical annotations . . . 11

3.3 Table type FbcObjective for flux balance models . . . 12

3.4 Table types Position and Layout for network graphics . . . 12

3.5 Table types Quantity and QuantityInfo for biochemical parameters . . . 12

3.6 Networks and relations . . . 14

3.7 Table for tool options . . . 15

4 SBtab conventions for metabolic state data **16**

4.1 Tables for a single metabolic state . . . 16

4.2 Tables for multiple metabolic states . . . 17

5 SBtab tables for sparse matrices and relations **18**

6 Software tools **18**

6.1 SBtab online tools . . . 19

6.2 Python code . . . 19

6.3 Visualisation in HTML . . . 19

6.4 Code and tools for R and Matlab . . . 20

7 Modelling tools and online resources **21**

7.1 Conversion to and from SBML . . . 21

7.2 Modelling tools supporting SBtab . . . 22

A Overview of the SBtab format **24**

A.1 Summary of SBtab rules . . . 24

A.2 General document and table attributes . . . 26

A.3 General table columns . . . 27

A.4 Table types in the SBtab default format . . . 27

B Predefined terms and recommended controlled vocabularies **33**

C Changes since SBtab version 1.0 **35**

1 Introduction

Spreadsheets and delimited text tables are the most utilised data formats in Systems Biology. They are easy to use and can hold various types of data. Tables can not only store omics data, but also metabolic network models described by lists of biochemical reactions. However, when tables are exchanged within scientific collaborations, modellers usually prefer tables that can be processed automatically, and the flexibility of spreadsheets can become a disadvantage. If table structures and nomenclature vary from case to case, parsing becomes laborious and new files require new parsers. Furthermore, different naming conventions – for instance, for biochemical compounds – make it hard to combine data, for instance metabolic network models and omics data produced by different researchers. Therefore, rules for structuring tables and for consistent naming and annotations can make tables much more useful as exchange formats in Systems Biology collaborations and for usage in software tools. The SBtab format comprises a set of conventions for data tables that are supposed to make tables easier and safer to work with. Let us start with a couple of examples. Then we continue with a more formal specification of SBtab version 1.2.

Example 1: Structure of a metabolic network model A stoichiometric metabolic model can be defined by a list of biochemical reaction formulae, specifying the substrates, products, and their stoichiometric coefficients. Such reactions can be listed in a single column of a spreadsheet, and additional information may be provided: each reaction can have a number or identifier (defined only within the model) and can be linked to an entry in the database KEGG Reaction [1]. Furthermore, reactions may be catalysed by enzymes, which relates them to certain genes. All information could be stored in the following table:

Reaction	Formula	KEGG ID	Gene symbol
R1	ATP + F6P <=> ADP + F16P	R00658	pfk
R2	F16P + H2O <=> F6P + Pi	R01015	fbp

where `ATP`, `F6P`, `ADP`, `F16P`, `H2O`, and `Pi` are element IDs for metabolites to be used in the model. Although the information is complete and unambiguous, the parser still has to recognise that the columns `Formula` and `KEGG ID` contain reaction formulae and identifiers in certain formats. If the column names and the syntax of the reaction formulae vary from table to table (e.g. <-> is used instead of <=>), parsing becomes tedious. In the SBtab format, the table would look a little more complicated, but is easy to parse automatically:

!!SBtab TableID='reaction_example' TableType='Reaction' SBtabVersion='1.2'			
!ID	!ReactionFormula	!Identifiers:kegg.reaction	!Gene:Symbol
R1	ATP + F6P <=> ADP + F16P	R00658	pfk
R2	F16P + H2O <=> F6P + Pi	R01015	fbp

In this table, elements highlighted by colours[1] have special meanings. The SBtab table differs from the original table in several ways: the first line[2] (containing only a single field and starting with !!) declares that the table is an SBtab table of the type Reaction and must therefore satisfy syntax rules for this table type.

A table attribute TableID must be provided, but can be chosen arbitrarily. However, it should adhere to the syntax conventions explained below (see section 3.2). The version 1.2 of the SBtab format is given as an optional argument. The following line contains the column headers. They start with the ! character, emphasising that they were not chosen *ad hoc* by the user, but stem from a controlled vocabulary. The predefined column headers do not contain whitespaces. The header `KEGG ID` has been replaced by the term !Identifiers:kegg.reaction. This may look complicated, but it allows parsers to retrieve further data from databases in a stable way[3]. The syntax of the reaction formulae is also uniquely defined. In particular, the

[1]For convenience, predefined SBtab entries will be colour-highlighted in this text. The colour highlighting is not a part of the SBtab format. The different colours have the following meaning. Table types and Column types defined by the SBtab format are listed in Table 1. Element IDs can be chosen *ad hoc* by the user; each of them needs to be defined by a table row. Element IDs have to be unique and consistent within a document, but may differ between documents. Reserved names are predefined in SBtab for recurrent mathematical expressions like "mean value". Official names, like the names used for databases, are defined by some other authority. `Free text` and other text including database IDs, numerical values, mathematical brackets, and operators is written in black.

[2] Please note that we distinguish between "lines" in a spreadsheet table or .tsv table, and "rows" within SBtab tables.

[3]The expression kegg.reaction is defined by the MIRIAM resources and used within SBtab. The URL of the KEGG database, defining the identifiers, may change in the future; however, KEGG's Miriam ID (provided by the the MIRIAM resources web service [2]) is guaranteed to remain stable in time.

element IDs of metabolites must not contain any whitespaces or special characters, which simplifies parsing and makes them suitable as variable names for computer models. The meaning of these element IDs can be defined by providing standardised names or database identifiers in a second table of type Compound. The compound element IDs will then serve as keys to rows of this table.

!!SBtab TableID='compound_example' TableType='Compound' SBtabVersion='1.2'		
!ID	!Name	!Identifiers:kegg.compound
F6P	Fructose 6-phosphate	C05345
ATP	ATP	C00002
ADP	ADP	C00008
F16P	Fructose 1,6-bisphosphate	C00354
H2O	Water	C00001
Pi	Inorganic phosphate	C00009
PEP	Phosphoenolpyruvate	C00074
AMP	AMP	C00020

Both tables together form an SBtab document describing a model. In practice, they can be stored in a single spreadsheet table or single table file, or in separate spreadsheet tables or separate table filed. The following example contains all necessary information to build a stoichiometric model in the SBML (Systems Biology Markup Language) format [3] (a software tool for conversion between SBtab and SBML is described in section 7.1):

!!!SBtab DocuemntName='Metabolic Model' DocuemntID='MetabolicModel' SBtabVersion='1.2'			
!!SBtab	TableID='reaction_example'	TableType='Reaction'	
!ID	!ReactionFormula	!Identifiers:kegg.reaction	!SBML:reaction:id
R1	ATP + F6P <=> ADP + F16P	R00658	r1
R2	F16P + H2O <=> F6P + Pi	R01015	r2
!!SBtab	TableID='compound_example'	TableType='Compound'	
!ID	!Name	!Identifiers:kegg.compound	!SBML:species:id
F6P	Fructose 6-phosphate	C05345	f6p
ATP	ATP	C00002	atp
ADP	ADP	C00008	adp
...	...	...	...
%/ That's a comment /%			

Here, to avoid to redundancy, we declare the SBtab version in another, optional line on top, starting with !!!. The tables are separated by (optional) empty lines for convenience. Also note the line surrounded by %/ .. /%: it's a comment line, which is ignored by the parser. Furthermore, we have added new identifiers (in the columns SBML:reaction:id and SBML:species:id) for Reaction and Compound entries to be used in SBML. Such extra names could be necessary if the original element IDs do not comply with SBML's rules for element identifiers.

Example 2: Table of kinetic constants In a second example, we specify numerical parameters, for example kinetic constants and metabolite concentrations that appear in a kinetic model. Each quantity can be related to a compound (e.g. a concentration), to a reaction (e.g. an equilibrium constant), or to several biological elements (e.g. to an enzyme and a compound, in the case of Michaelis-Menten constants). As in the previous example, these elements can be specified by unique identifiers, e.g. KEGG compound or reaction identifiers. Furthermore, each quantity has a value and a physical unit. In the SBtab format, we arrange this information in a table of type Quantity. Each row contains all information about one of the quantities:

!!SBtab TableID='quantity_example' TableType='Quantity' TableName='Some kinetic constants'					
!ID	!QuantityType	!Reaction:Identifiers:kegg.reaction	!Compound:Identifiers:kegg.compound	!Value	!Unit
keq_R1	equilibrium constant	R01061		0.156	dimensionless
kmc_R1_C1	Michaelis constant	R01061	C00003	0.96	mM
kic_R1_C1	inhibition constant	R01070	C00111	0.13	mM
con_C1	concentration		C00118	0.203	mM
...	...	...	...	...	...

In this example, we have added another optional attribute to the first line, TableName, which can be chosen freely and does not underlie any syntax restrictions. The first two columns specify a name and a type for each quantity. The quantity types (substrate catalytic rate constant, equilibrium constant etc.) are not chosen *ad hoc*, but stem from the Systems Biology Ontology (SBO) [4]. This ensures a unique spelling and allows software to retrieve definitions and further information from the SBO web services. The biological elements (in this case, reactions, compounds, or both) are specified in the following two columns by unique identifiers from the KEGG database. Columns with human-readable names, or identifiers from other databases, could be added. Unnecessary fields remain empty. The column name Value – like some other mathematical terms – is defined for SBtab (arbitrary values in this example). Unit names are defined as in SBML (see below). If the table is used together with a metabolic model, we can use compound and reaction identifiers from the model instead of the Identifiers.org annotations [5]. In this case, the table would read:

!!SBtab TableID='quantity_example_2' TableType='Quantity'					
!ID	!QuantityType	!SBML:reaction:id	!SBML:species:id	!Value	!Unit
Par_1	equilibrium constant	r1		0.156	dimensionless
Par_2	Michaelis constant	r1	atp	0.96	mM
Par_3	inhibition constant	r1	atp	0.13	mM
Par_4	concentration		atp	1.5	mM
...	...	...	...	...	...

This table, together with a stoichiometric model and a choice of standardised rate laws (like the modular rate laws [6]) completely defines a kinetic metabolic model.

Example 3: A table with metabolome data As a last example, let us consider a table with metabolome time series data. For the sake of simplicity, only two metabolites (rows) and measured samples (columns) are shown:

!!SBtab TableID='quantity_example_3' TableType='QuantityMatrix'				
!Compound	!Identifiers:obo.chebi	t = 0 s	t = 0.5 s	..
Glucose	CHEBI:17234	1.1	1.2	..
Fructose	CHEBI:15824	1.4	0.9	..
..	..	..	..	..

Tables of this sort can be also be used for other kinds of omics data. In this example, the headers of data columns (e.g., `t = 0 s`) do not follow a specific syntax and contain relevant information (time point and time unit). We shall see below how such information can be provided in SBtab in a more structured manner. Note that this "QuantityMatrix" table does not contain an ID column, but just columns to specify the identity of the rows.

In this specification for SBtab version 1.2, we first introduce the general SBtab format and conventions for all kinds of data (Section 2). Then we define predefined table types and conventions specific to systems biology models and data, for example, a syntax for reaction formulae (Section 3). Finally, we present available software and online tools for parsing, validating, and converting SBtab files (Section 6). The appendix gives an overview of all predefined table types A. Appendix B lists recommendations about controlled vocabularies and database resources to be used in SBtab files. Changes after SBtab version 1.0 are listed in appendix C.

2 General form of the SBtab format

2.1 SBtab format and biological data types

Conventions in SBtab SBtab comprises a list of conventions for the structure, nomenclature, syntax, and annotations in tables describing biochemical network models, kinetic parameters, and dynamic data. As a general format, it contains

1. General rules for table structure and syntax used in table fields.
2. Rules for usage of names, element IDs, and database identifiers used for annotation.
3. Syntax rules for mathematical formulae
4. A mechanism for specifying the table types and the columns appearing in these tables. Table types are defined through a special Definition table. By modifying this schema, new column or table types can be declared to extend the format.

Conventions for biological models and data While the general rules apply to all kinds of data, the current version of SBtab is tailored for describing the structure of biochemical network models and the biochemical quantities therein. More specifically, as a format for biological modelling it provides

1. Defined table types for different kinds of information, each with possible columns with defined names and data types. The predefined table types, as defined by a default Definition table, are listed in in Table 1, described in section , and a more detailed description is given in the appendix.
2. A syntax for biochemical element annotations pointing to databases or ontologies.
3. Naming rules for biochemical quantities to specify the quantities, physical units, and mathematical terms (such as Mean for mean values).
4. A syntax for reaction sum formulae, kinetic rate laws.

This section describes SBtab's general format conventions, referring to biochemical use cases as examples. Specific conventions for biological models and data are discussed in the following section 3.

2.2 SBtab documents and SBtab tables

SBtab elements and IDs An SBtab document consists of one or several tables that refer to a common model or related data sets. Each table row is called an *SBtab element*, with attributes given by the different fields in this row. An SBtab element can carry a element ID, specified in an ID column. To avoid ambiguities, Element IDs, table IDs, and the document ID should be unique across the entire document. For instance, a Compound table contains the column !ID, and the elements from this column define compound element IDs to be used in the other tables. Table types and columns appearing in these tables are defined in a Definition files, which can be customised.

SBtab files: file format and tables SBtab documents consist of one or more SBtab tables, stored in a spreadsheet files or delimited text file. All SBtab tables and further information can be stored in a single (e.g. tab-delimited or spreadsheet) table. Alternatively, different tables may be provided in separate spreadsheets (or separate tab-delimited files).

Comments Lines starting with the % character are treated as comment lines and are ignored by the parser. It is good practice to start commnt lines with %/ and end them with /%. Moreover, comments that concern a specific table should be placed directly above this table; comments that concern a specific table row should be placed directly above this row. Empty lines can be used for visual separation and will be ignored by the parser.

Document attributes A document can begin with an optional attribute line starting with !!!SBtab. A document can start with an (optional) Document attribute line, starting with !!!SBtab, followed by the document attributes in the syntax *attribute name*='*attribute value*'. For example, a date can be specified by `Date='2020-01-01'`. Most importantly, it must contain an attribute Document, which is an identifier via which the document can be referred to (e.g. in the Document attribute in table attribute lines, which defines to which document a table belongs). Optionally, a DocumentName and DocumentType can be declared, as well as the Date. Different attributes must appear in the first cell on the left and must be separated by single whitespaces. Simple quotation marks (UTF-8 code 27) should be used for the attribute value, (while double quotes " may be used to enclose cells in comma-delimited table files). Some other quotation marks are tolerated by the parser, but their use is not recommended. For recommended document attributes, see section A.2.

Table attributes Each table starts with a (mandatory) line containing the table attributes, resembling the document attribute line. The first field contains the table header, starting with !!SBtab and followed by the table attributes in the syntax *attribute name*='*attribute value*'. Again, other quotation marks are tolerated by the parser, but are not recommended. A mandatory attribute is TableID, which must be a unique identifier of the table within the document. It can be chosen arbitrarily, but may not start on a digit and must not contain special characters or whitespaces. the TableID is missing, our software tools will generate a default identifier and add it to the table. Another mandatory attribute is the TableType. Its value can be one of the default SBtab table types or a customised table type declared in an accompanying Definition table. Several tables in a document may have the same type, but their table dentifiers (indicated by the table attribute TableID) must be unique within the document. An optional table attribute is TableName, defining an arbitrary name for the table, without any syntax restrictions. We recommend to list these three attributes in the order TableID, TableType, TableName. Other recommended table attributes are listed in section A.2. Any other table attributes can be added, as long as their names are valid element IDs (no whitespaces or special characters, not starting with a numerical character). The same attributes may appear in tables and in the document. However, as a rule of good practice, document attributes (e.g. Date) should not be repeated in individual tables, unless they provide non-redundant information.

Table columns Each table type can have a number of particular columns, as specified in the Definition table. Column names start with a ! character and must be defined in Definition table used during parsing (see section 2.6). As an exception, there can be column headers that are not defined in the Definition table, but point to SBtab elements in another table. For instance, if a table contains a row with key SampleID, defining the meaning of the term "SampleID" for this document, then `!>SampleID='Sample 1'` can used as a valid table attribute in any table within that document. Columns without ! or may exist in the file, but are ignored during parsing.

Transposed tables If a table contains just a few rows, but many columns, it is convenient to write this table with rows and columns flipped. This can be declared by the table attribute TableOrientation='Transposed'. For example, our first example table from the introduction section could also be written as

!!SBtab TableID='reaction_example' TableType='Reaction' TableOrientation='Transposed'		
!ID	R1	R2
!ReactionFormula	ATP + F6P <=> ADP + F16P	F16P + H2O <=> F6P + Pi
!Identifiers:kegg.reaction	R00658	R01015
!Gene:Symbol	pfk	fbp

Header line, column names, and definition table The second line of a table contains the column headers. Columns whose headers start with a ! are treated as SBtab columns and must adhere to the SBtab rules. Other columns can contain arbitrary content. SBtab has a number of predefined table types that can hold different kinds of data. Each table type has a number of mandatory or optional columns with specific properties. An overview is given below and in the appendix. However, users can also define their own table types and corresponding columns. This definition must be provided by the user in the form of a special Definition table (as described below). The order of the columns can be chosen freely, but it is good practise to set ID as the first column.

SBtab element IDs SBtab elements (represented by table rows) can carry unique element IDs, the arbitrary element names used in a data set or model. Element IDs must be unique, i.e. each element IDs must be declared only once in a document. Unique keys for table rows can be defined in a column ID. It is good practise to make the first column the ID column. The keys should be unique across the entire SBtab document (although this is not checked by the parser) and must satisfy the following syntax rules: they must start with a letter and may not contain spaces or the special characters ":", ".".

JSON strings SBtab supports the usage of JSON syntax within the table columns. This can be used, for example, to provide structured biochemical annotations (see section 3.2 for an example). Please note that structured annotations are currently not supported by the SBtab ↔ SBML converter. The syntax for a JSON column in SBtab is {"A":"X", "B":"X"}. In the strings, make sure to employ ASCII supported quotation marks. We discourage the use of JSON strings describing data structures with multiple levels.

2.3 Use of comment lines

For convenience, tables can be described by comment lines starting with %. Writing comment lines of the form `%/TEXT/%` is recommended as good practice. We recommend to put comments about a table above the table, and comments about a table row above the table row. Putting empty lines for convenience, and using again an example from section , we obtain:

!!!SBtab DocumentName='Example document' SBtabVersion='1.2'			
%/ Description of Table reaction_example /%			
!!SBtab TableID='reaction_example' TableType='Reaction'			
!ID	!ReactionFormula	!Identifiers:kegg.reaction	!SBML:reaction:id
%/ Description of element R1 /%			
R1	ATP + F6P <=> ADP + F16P	R00658	r1
%/ Description of element R2 /%			
R2	F16P + H2O <=> F6P + Pi	R01015	r2
%/ Description of Table reaction_example /%			
!!SBtab TableID='compound_example' TableType='Compound'			
!ID	!Name	!Identifiers:kegg.compound	!SBML:species:id
F6P	Fructose 6-phosphate	C05345	f6p
ATP	ATP	C00002	atp

2.4 Syntax rules for mathematical expressions

Mathematical formulae We recommend to write mathematical in the syntax adopted by SBML (a syntax also shared by python and MATLAB), with the symbols +,−,∗,/, and ˆ as well as round brackets.

2.5 File formats

To ensure consistency between character-delimited text files and spreadsheet files, we propose a number of rules for good practice:

- **UTF8 encoding** If possible, the UTF8 encoding should be chosen. Some of the software tools may run into errors if other encodings are employed.
- **Documents** In character-delimited text files (.tsv or .csv), a document can either be stored in several files with the filenames *basename_tablename.extension*, or tables are concatenated vertically, each preceded by an attribute line (starting with !!), and stored in a single table file. In the latter case, the document may be preceded by a document attribute line starting with !!!SBtab....
- **Delimiters in .tsv or .csv files** In character-delimited files, it is recommended to use the filename extension ".tsv" for tab-delimited files and the extension ".csv" for comma-delimited files. However, other delimiters (comma or semicolon) are accepted, and the python parser will try to guess the delimiter in any case. We recommend the usage of .tsv files.

- **Special characters** If table cells contain special characters that are also used as cell delimiters (e.g. commas), the file must be provided in a form that excludes ambiguities (e.g. in the case of a comma-delimited table containing commas with its fields, all cells must additionally be marked by double quotation marks (`".."`). For details, see the csv format specification at `https://www.ietf.org/rfc/rfc4180.txt`.
- **Excel datafiles** Aside from character-delimited file formats, SBtab documents may also be stored as Microsoft Excel documents with the .xlsx extension.

Filenames The SBtab format as such does not impose any restrictions on filenames, nor does it require a specific filename extension. SBtab files stored as excel sheets, for instance, will have the extension `.xlsx`. However, the SBtab online tools (and the python programs behind it) have a certain convention for filenames and filename extensions. When an SBtab document is exported to several delimited text files, the filenames will be chosen according to the scheme `[SBTAB DOCUMENT NAME]_[TABLE TYPE].tsv` (in the example of a tab-delimited file) or, in case of ambiguities `[SBTAB DOCUMENT NAME]_[TABLE TYPE]_[TABLE NAME].tsv`.

Filename extensions Regarding filename extensions, the python implementation of SBtab supports comma-delimited and tab-delimited tables, as well as excel spreadsheet files (xlsx). By default, the python code exports tab-delimited files and uses the filename extension `.tsv`. Some versions of LibreOffice expect the filename extension ".csv" also for other delimiters used. In case of conflict, users may have to rename their files. When importing a table, the code tries to determine whether commas or tabs are used as delimiters. When using commas as delimiters, users have to make sure that no commas are used elsewhere in the table (or that all elements are given in double quotes).

2.6 Defining an SBtab format

Definition table Users can define their own table types and corresponding columns. For usage in the online tools or in the Python code, this definition can be provided by the user in the form of a special Definition table. The default table (containing the predefined table and column types) is available on the SBtab website[4] Note that, when using a new Definition table, the predefined Definition table will be completely overridden, so any tables and columns to be used (also the predefined ones) must be listed in the new table. The typical format of a Definition table is shown below.

!!SBtab TableID='definition' TableType='Definition'					
!ComponentName	!ComponentType	!IsPartOf	!linksElementID	!Format	!Description
%/ Table type 'Compound' /%					
Name	Column	Compound	False	String	Compound name
SBML:species:id	Column	Compound	False	String	SBML species ID of compound
%/ Table type 'Reaction' /%					
SBML:reaction:id	Column	Reaction	False	String	SBML ID of reaction
ReactionFormula	Column	Reaction	False	String	Reaction sum formula
Enzyme	Column	Reaction	True	String	Enzyme catalysing the reaction
...	...	...	...	..	..

The Format column defines which type of entries a column can contain. Possibilities are string, Element ID (name of SBtab element, as defined in one of the SBtab tables), integer, float, and Boolean (with possible values True and False. Structured string formats (e.g., the syntax for reaction sum formulae) are not formally defined and should be informally explained in the Description column. The linksElementID column states whether the component refers to a element ID of an SBtab element, defined in the ID column of another table. It is good practice to start the description of each table type with a comment line defining the table name, as in the example above.

How to define an SBtab format An SBtab format is defined by a schema table of the TableType Definition (see section 2.6), which declares all allowed document types, table types, and column types. The following rules help make tools interoperable and make files better readable for humans

[4] `https://www.sbtab.net/sbtab/static/documents/definitions.tsv`

- The format should be defined by a `[FORMATNAME]_Definition.tsv` file, stating all mandatory tables and columns. This file describes the schema and should be provided in a public repository.
- In designing the format, existing table and column types (from SBtabs default Definition table should be reused wherever possible
- Syntax recommendations from the SBtab specification should be applied wherever possible.

2.7 MIRIAM-compliant models

The MIRIAM rules for computational models [7] have been established to guarantee that published models contain complete and unambiguous information, and that results from the models can be verified. Note that MIRIAM-compliance also involves criteria that cannot be ensured by the file structure alone, but are related to how the model was made, and to the existence of a reference publication (which may or may not exist for a given SBtab file). (i) The encoded model structure must reflect the biological processes described by the reference description. (ii) The model must be instantiable in a simulation: all quantitative attributes must be defined, including initial conditions. (iii) When instantiated, the model must be able to reproduce all results given in the reference description within an epsilon (algorithms, round-up errors).

However, to allow users to satisfy some of the MIRIAM requirements, SBtab contains document attributes for information that is mandatory for MIRIAM-compliance. These must be given in the attribute line of the SBtab document in question, or in the attribute lines of at least one tables belonging to the document (i) ReferenceDescription (ii) DocumentName (iii) ReferenceCitation (complete citation, unique identifier, unambiguous URL). The citation should identify the authors of the model. (iv) ModelCreators (name and contact information for model creators) (v) ModelCreationTime (The date and time of model creation and last modification) (vi) TermsOfDistribution (link to a precise statement about the terms of it's distribution).

While the first list of points is specific to computational models, the points in the second list can refer to any type of data, and we encourage the use of these document attributes generally.

3 SBtab formats for biological models and data

A main purpose of SBtab is to describe biological data and models. To this aim, SBtab predefines a number of relevant table types and adopts a number of syntax conventions. An overview of the predefined table types is given in Table 1. They are defined in the default Definition table, which can be customised by the user.

3.1 Biochemical networks and models

As in example 1 (in the introduction section), biochemical networks consist of biochemical entities (e.g. metabolites or proteins) and reactions or interactions between them. The tables describing these entities (table types Reaction, Compound, Compartment, Enzyme, Regulator, and Gene) have to satisfy the following rules.

- **Entities** In tables describing biochemical entities (Compound, Enzyme, Gene, Regulator, Compartment), each row has to contain (i) a element ID as the primary key (in the column !ID) and (ii) at least one entry specifying the entity, like !Name or !Identifiers:*DB*. If a column shares the type of the table (e.g. a Compound column in a Compound table), it can be considered a primary key, that is, its elements should be unique and it should appear as the first column in the table. Optional columns - which may depend on the kinds of entities - are listed in Table A.4.

- **Reactions** A Reaction table lists chemical reactions, possibly with information about the corresponding enzymes, their kinetic laws, and their genetic regulation. It must contain at least one of the following columns: !ReactionFormula, !Identifiers:*DB*; optional columns are listed in Table 13. For an example, see example 1 in the introduction.

- **Enzymes, genes, and regulators** The connection between chemical reactions, the enzymes catalysing the reactions, and the genes coding for the enzymes can be complicated, but in many cases, there is a one-to-one relationship. In SBtab, there are different ways to express this relationship. Information about enzymes or genes and their regulation can be stored in a Reaction table if there is a one-to-one relationship between reactions, enzymes, and possibly genes. Otherwise, it is stored in separate tables Enzyme and Gene and the tables are interlinked *via* the columns !Enzyme (in table Reaction) and !Gene (in table Enzyme) or !TargetReaction (in an Enzyme table) and !GeneProduct (in a Gene table).

- **Flux balance models and network layout** The table FbcObjective contains information on objective functions for Flux Balance Analysis (compatible with the SBML fbc package). The table Layout contains information on objective functions for Flux Balance Analysis (compatible with the SBML layout package).

Reaction formulae Chemical reactions can be described by reaction formulae (column !ReactionFormula in table Reaction; specifying the reactants, their stoichiometric coefficients, and possibly their localisation). The reaction arrow is denoted by <=>. Stoichiometric coefficients refer to substance amounts, not concentrations (this matters in the case of transport reactions). Stoichiometric coefficients of 1 are omitted; general stoichiometric coefficients, given by letters (e.g. `n`) are not allowed. If possible, the reaction formula should represent the actual stoichiometries experienced by the enzyme (i.e. `A <=> 2 B` rather than `0.5 A <=> B`). Substrates and products are given by element IDs, which must be defined in a Compound table. The order of substrates and the order of products are arbitrary; however, comparison of formulae is eased by using an alphabetical order. Localisation in compartments can be described as follows:

- Reaction in the default compartment: `A + 2 B <=> C + D`
- Reaction in a specific compartment: `[comp1]: A + 2 B <=> C + D`
- Transport reaction: `A[comp1] + 2 B[comp1] <=> C[comp2] + D[comp2]`

In the example, A, B, C, and D are compound element IDs, and comp1 and comp2 are compartment element IDs. The reversibility of reactions is not encoded in the reaction sum formula, but defined by an extra column !IsReversible in the Reaction table.

Name	Contents	Usage
Compound	Names, IDs, properties of compounds	model structure
Enzyme	Names, properties of enzymes	model structure
Protein	Names, properties of proteins	model structure
Gene	Names, properties of genes	model structure
Regulator	Names, properties of gene regulators	model structure
Compartment	Names and IDs of compartments	model structure
Reaction	Chemical reactions	model structure
ReactionStoichiometry	Stoichiometric coefficients	model structure
FbcObjective	Flux Balance Analysis objective	model configuration
Position	Glyph positions for graphical network layout	model layout
Layout	Network layout description	model layout
Quantity	Individual data for model parameters	quantitative data
Measurement	Measurements or samples	quantitative data
QuantityMatrix	Data matrices	quantitative data
QuantityInfo	Definition of parameter types	parameter priors and ranges
Relation	Graphs and relations	network data
Config	Tool configuration files	software configuration
Definition	Define custom column types, etc.	customising SBtab

Table 1: Overview of predefined table types in SBtab. The table types Compound, Enzyme, Protein, Gene, Regulator, Compartment, and Reaction, and ReactionStoichiometry describe biochemical network models. Some additional information, specifically used in SBML can be stored the table types FbcObjective, Position, and Layout. The table types Quantity, Measurement, QuantityMatrix, and QuantityInfo are used to describe quantitative and experimental data. Relation tables can describe general networks and pairwise relations. Definition tables are used to specify SBtab schemas, while Config tables are used to specify tool options.

Alternatively, a ReactionStoichiometry table can be used to describe the stoichiometric structure of a reaction network. Here is an example. A normal SBtab Reaction table for glycolysis as a net reaction

!!SBtab TableID=’rs_example_1’ TableType=’Reaction’		
!ID	!Name	!ReactionFormula
glycolysis	Glycolysis	Glc + 2 ATP + 4 ADP + 2 Pi + 2 NAD <=> 2 Pyr + 4 ATP + 2 ADP + NADH

can automatically be translated into a format describing the stoichiometric coefficients:

!!SBtab TableID=’rs_example_2’ TableType=’ReactionStoichiometry’				
!ID	!Stoichiometry	!Substrate	!Product	!Location
glycolysis	1	Glc		Extracellular
glycolysis	2	ATP		Cytosol
glycolysis	4	ADP		Cytosol
glycolysis	2	Pi		Cytosol
glycolysis	2	NAD		Cytosol
glycolysis	2		Pyr	Extracellular
glycolysis	4		ATP	Cytosol
glycolysis	2		ADP	Cytosol
glycolysis	1		NADH	Cytosol

The new parseable SBtab table also offers the possibility of manually assigning spatial information to the individual reaction products and educts. This is practical, since clustered reactions in SBML and SBtab do not offer such a diverse localisation for different reaction entities. In the example, we are implying the information that glucose is imported into the cell and pyruvate is exported.

Kinetic rate laws and enzyme regulation Kinetic rate laws (column KineticLaw in Reaction table) are given as mathematical formulae in the syntax adopted by SBML (a syntax also shared by python and MATLAB), with the symbols +,−,∗,/, and ˆ as well as round brackets. Enzyme regulators are given in a column Regulator in the same table.

3.2 Biochemical annotations

Naming and specification of biological entities Biochemical elements can be described by element IDs, official names, or database identifiers (IDs). The biochemical meaning of SBtab elements can be declared by different columns:

- !Name contains official names (it is good practice to use names from the suggested databases). Several names can be listed in one field, separated by "|". To declare from which database a name has been taken, the name can also be written as *DB*:*name*.
- !Identifiers:*Identifiers* contains IDs from a specified database. Annotations with database IDs follow the scheme defined by Identifiers.org [5] (data collection and ID).

Element identifiers, names, and annotations In columns containing database IDs, the column name (!Identifiers:*Identifiers*) specifies the database by a name (to be used in column names, IDs etc.) and an URI. We suggest to use preferably the databases listed in the Miriam file (see Table 25). Sometimes, elements may be characterised redundantly: e.g. the reaction catalysed by an enzyme, given in an Enzyme table, can be given by both element ID and database ID. In case of conflict, the information derived from the element ID (i.e. the database ID listed in the Reaction table) has higher priority.

Annotating biochemical elements with database identifiers Biochemical elements are annotated with database IDs listed in special identifier columns. An Identifiers column contains annotations from one web resource, at most one annotation per element, and without qualifiers. The column item and the referenced ID are assumed to be linked by an "is" relationship (and not, for instance, "version of", which can exist in SBML annotations). A table can contain several Identifiers columns, which must refer to different data resources.

!!SBtab TableID='annotation_example' TableType='Compound'			
!ID	!Identifiers:obo.chebi	!Identifiers:kegg.compound	...
water	CHEBI:15377	C00001	...
ATP	CHEBI:15422	C00002	...
phosphate	CHEBI:18367		...

To translate an element like CHEBI:16865 into a valid Identifiers.org URI, `http://identifiers.org/` is concatenated with the data collection mentioned after !Identifiers: in the header (e.g. obo.chebi) and with the column item, separated by a slash[5]. For instance, the first annotation entry in the table above would be resolved to `http://identifiers.org/obo.chebi/CHEBI:15377`.

To store more complex annotations, including bioqualifiers and several annotations involving the same resource, we propose a column called !MiriamAnnotation. The entire URN-encoded URN is stored in the column element. Each column element contains a JSON string listing all annotations; each annotation consists of the bioqualifier and the MIRIAM URN (for details, see the conventions used in SBML). SBtab supports the qualifiers (like "Is", "VersionOf") defined in the MIRIAM resources [8]. They can be used within table cells (in the syntax "*qualifier Identifiers*". If no qualifier is given (empty string ""), the qualifier "unknown" will be assumed by default.

!!SBtab TableTitle='Ex 8 - Miriam Annotation' TableType='Compound'	
!ID	!MiriamAnnotation
water	{"":"urn:miriam:obo.chebi:CHEBI%3A15377","":"urn:miriam:kegg.compound:C00001"}
ATP	{"bqbiol:hasPart":"urn:miriam:obo.chebi:CHEBI%3A18367"}

The above columns corresond to the elements listed in the first column of the table. It is also possible to annotate elements of other columns. For instance, a column named "Enzyme" could be accompanied by columns "!Enzyme:MiriamAnnotation".

[5]The elements from the column have to be translated into a URN-encoded form (as described in the URN specification): for instance, the colon in the identifier CHEBI:16865 has to be replaced by the string "%3A" to create the URN obo.chebi:CHEBI%3A16865.

3.3 Table type FbcObjective for flux balance models

Flux balance analysis models require additional data about the flux objective and, possibly, about the mapping between chemical reactions and the genes coding for the catalysing enzymes. In SBML, this information is handled by the SBML 3 fbc package (for details, see section 7.1). In SBtab, information about flux objectives is given by tables of type FbcObjective:

!!SBtab TableTitle='Ex - FbcObjective' TableType='FbcObjective' TableName='FBC Objective'				
!ID	!Type	!Active	!FluxObjectiveCoefficient	!FluxObjectiveReaction
obj	maximize	True	1.0	R_biomass_mm_1_no_glygln

Gene associations are stored in a Gene table, possibly in addition to other existing data:

!!SBtab TableTitle='Ex Fbc Gene TableType='Gene' TableName='FBC Gene'				
!ID	!SBML:fbc:ID	!SBML:fbc:Name	!SBML:fbc:GeneProduct	!SBML:fbc:Label
G_100039108	G_100039108	100039108	True	100039108
G_100041375	G_100041375	100041375	True	100041375

3.4 Table types Position and Layout for network graphics

To represent a biochemical network as a network graphics, the positions of compounds and reactions in the graphics must be defined. This can be done by a Position table of the following form:

!!SBtab TableTitle='Ex Simple network graphics' TableID='Network layout' TableType='Position'			
!Element	!PositionX	!PositionY	!Label
AcetylCoA	0.36	0.22	AcCoA
Fumarate	0.32	-0.0	Fum

A more comprehensive description of network graphics is given by tables of the type !Layout, which implement the functionality of the SBML 3 layout package (see section 7.1). Here is an example:

!!!SBtab DocumentName='SBML layout example' SBtabVersion='1.2'									
!!SBtab TableID='layout' TableType='Layout'									
!ID	!ModelEntity	!SBML:compartment:id	!SBML:reaction:id	!SBML:species:id	!CurveSegment	!SBML:X	!SBML:Y	!SBML:width	!SBML:height
Layout_1	LayoutCanvas							400.0	220.0
CompartmentGlyph_1	Compartment	Compartment_1				5.0	5.0	390.0	210.0
SpeciesGlyph_1	Species			Species_1		80.0	26.0	240.0	24.0
TextGlyph_01	SpeciesText			Species_1		92.0	26.0	228.0	24.0
ReactionGlyph_1	ReactionCurve		Reaction_1		Start	165.0	105.0		
ReactionGlyph_1	ReactionCurve		Reaction_1		End	165.0	115.0		
SpeciesRefGlyph_1	SpeciesReferenceCurve		Reaction_1	SpeciesGlyph_1	Start	165.0	105.0		

3.5 Table types Quantity and QuantityInfo for biochemical parameters

Numerical data (e.g. for time series or kinetic parameters) can be stored in tables and be linked to model elements via the latters' element IDs. SBtab provides different table types for numerical data. Tables of type Quantity describe individual physical or biochemical quantities, for instance, kinetic parameters in a network model. These quantities can be linked to one entity, one reaction or enzyme, or both. If a quantity table contains several values for the same quantity, they appear in separate rows (for possible descriptions of provenance, see Table 12). Tables of type QuantityMatrix serve to describe data matrices, where information about samples can be provided in a Measurement table. Finally, the table type QuantityInfo serves to define general properties of physical parameters, for example typical ranges.

Table type Quantity for individual data values Tables of type Quantity describe single physical or biochemical quantities (e.g. individual kinetic constants). A quantity is defined by a type, a unit, possibly biochemical entities to which it refers, possibly a localisation, and possibly experimental or physical conditions. The columns contain the defining properties (e.g. unit, conditions, etc.) and their values. Quantities can refer to a compound, an enzyme or reaction, or a combination of them. For instance, a concentration refers to a substance, while a k^{M} value refers to a metabolite and an enzyme. If there is a one-to-one relationship between reactions and enzymes, the k^{M} value can also be assigned to a compound/reaction pair or a compound/enzyme pair. Let us consider again example 2:

!!SBtab TableID='quantity_example_2' TableType='Quantity'

!ID	!QuantityType	!Reaction:Identifiers:kegg.reaction	!Compound:Identifiers:kegg.compound	!Value	!Unit
keq_R1	equilibrium constant	R01061		0.0984	dimensionless
kmc_R1_C1	Michaelis constant	R01061	C00003	0.96	mM
kic_R1_C1	inhibition constant	R01070	C00111	0.13	mM
con_C1	concentration		C00118	0.203	mM

To specify the parameters of a model, we refer to Reaction and Compound elements by element IDs rather than by resource IDs. In this form, the above example becomes

!!SBtab TableID='quantity_example_3' TableType='Quantity'

!ID	!SBO:Identifiers:obo.sbo	!Reaction	!Compound	!Value	!Unit
kcrf_R1	SBO:0000320	R1		200.0	1/s
keq_R1	SBO:0000281	R1		0.0984	dimensionless
kmc_R1_C1	SBO:0000027	R1	C1	0.96	mM
kic_R1_C2	SBO:0000261	R1	C2	0.13	mM
con_C3	SBO:0000196		C3	0.203	mM

This example shows that quantity types can be specified by identifiers from the Systems Biology Ontology (SBO) in a column !SBO:Identifiers:obo.sbo.

A Quantity table can also store state-dependent quantities like concentrations, expression levels, or fluxes, like in the following example.

!!SBtab TableID='quantity_example_4' TableType='Quantity'

!ID	!Compound	!Condition	!SBO:concentration	!Unit
con_C1_wt	C1	wildtype	0.2	mM
con_C2_wt	C2	wildtype	1	mM
con_C3_wt	C3	wildtype	0.1	mM
con_C1_mu	C1	mutant	0.1	mM
con_C2_mu	C2	mutant	0.5	mM
con_C3_mu	C3	mutant	0.1	mM

Table type Measurement for additional information on measurement values In the previous example, it would obviously be good to store time point and time unit separately instead of merging them in the column header. This can be realised as follows:

!!SBtab TableID='qm_example_3' TableType='QuantityMatrix'

!ID	!Identifiers:obo.chebi	!>Sample:t0	!>Sample:t1
Glucose	CHEBI:17234	1.1	1.2
Fructose	CHEBI:15824	1.4	0.9

!!SBtab TableID='measurement_example_2' TableType='Measurement' TableName='Sample'

!ID	!Time	!Unit	
t0	0	s	
t1	0.5	s	

The column header !>Sample:t0 in the first table refers to the row t0 in the second table. Since element IDs must be unique within SBtab documents, the column header could also simply be called !>t0. Using the same mechanism, a table with mean values and standard deviations for all measured numbers can be implemented as follows:

!!SBtab TableID='qm_example_3' TableType='QuantityMatrix'

!ID	!Identifiers:obo.chebi	!>TP:t0:mean	!>TP:t1:mean	!>TP:t0:std	!>TP:t1:std
Glucose	CHEBI:17234	1.1	1.2	0.5	0.5
Fructose	CHEBI:15824	1.4	0.9	0.5	0.5

!!SBtab TableID='qm_example_3' TableType='Measurement' TableName='TP'

!ID	!Time	!Unit	!ValueType		
t0:mean	0	s	Mean		
t0:std	0	s	Mean		
t1:mean	0.5	s	Std		
t1:std	0.5	s	Std		

Table type QuantityMatrix for data matrices Biological data often have the form of matrices. As an example, consider a small 2×2 matrix containing metabolite concentrations for two time points and two metabolites. It can be expressed by the following SBtab table.

!!SBtab TableID='qm_example' TableType='QuantityMatrix'		
!Time	Glucose	Fructose
0.0	1.1	1.4
0.5	1.2	0.9

The headers of the data columns are not defined headers starting with "!", but simple strings. Therefore, they are not formally controlled by SBtab. Annotating these columns, e.g., by adding ChEBI Identifiers to specify the metabolites, is not directly possible. Moreover, the time points have no keys to which other tables could refer. In an alternative solution, the column headers are controlled and point to rows of another table with table name "Concentration", in which the ChEBI Identifers are given:

!!SBtab TableID='qm_example_1' TableType='QuantityMatrix'			
!ID	!Time	!>Concentration:Glucose	!>Concentration:Fructose
T0	0.0	1.1	1.4
T1	0.5	1.2	0.9

!!SBtab TableID='Concentration' TableType='Quantity' TableName='Concentration'			
!ID	!Identifiers:obo.chebi	!QuantityType	!Unit
Glucose	CHEBI:17234	concentration	mM
Fructose	CHEBI:15824	concentration	mM

Note that references to other tables point to TableID, not TableName. Now let us consider data tables in which time points are represented by columns. A similar scheme can be used in this case:

!!SBtab TableID='qm_example_2' TableType='QuantityMatrix'			
!ID	!Identifiers:obo.chebi	t = 0 s	t = 0.5 s
Glucose	CHEBI:17234	1.1	1.2
Fructose	CHEBI:15824	1.4	0.9

Table type QuantityInfo for general properties of physical quantities Biochemical parameters, for example physiological metabolite concentrations, have typical values that can be described by prior probability distributions and ranges. Such information can be stored in a QuantityInfo table. The following example can be used to define prior distributions and feasible ranges for different quantities in parameter balancing [9]. For further details about this table type, see www.parameterbalancing.net

!!SBtab TableID='PP' TableType='QuantityInfo'							
!QuantityType	!Symbol	!Unit	!Element	!PriorMedian	!PriorStd	!LowerBound	!UpperBound
Michaelis constant	KM	mM	Reaction/Species	0.1	1	0.000001	1000
activation constant	KA	mM	Reaction/Species	0.1	1	0.0001	100
inhibitory constant	KI	mM	Reaction/Species	0.1	1	0.0001	100
equilibrium constant	Keq	dimensionless	Reaction	1	1.5	0.0000000001	100000000
concentration	c	mM	Species	0.1	1.5	0.000001	1000

3.6 Networks and relations

Table type Relation for pairwise relations The table type Relation is used to define pairwise links between objects. Each link ("relationship") can have a type and a numerical value. A Relation table can, for instance, be used to define a sparse matrix (by listing the row, column, and numerical value of each non-zero element), a directed graph (by listing the edges between nodes of one type), the stoichiometric matrix of a reaction network (by listing pairs of reactions and compounds their with stoichiometric coefficients), or a gene regulatory network (by listing the actions of transcription factors on gene promoters). In particular, Relation tables can be used to link SBtab elements between tables and, thus, to create SBtab documents that have the form of a relational database.

!!SBtab TableID='relationship_example' TableType='Relation'			
!From	!To	!Relation	!Value
A	A	regulates	1
A	B	regulates	-1
B	A	regulates	1
B	C	regulates	2
C	D	regulates	1

Sparse matrices in the SBtab format A quantitative pairwise relationship can also be seen as a sparse matrix. For convenience, SBtab provides four different ways to describe such matrices, with the following formats:

!!SBtab TableID='sm_example_1' TableType='SparseMatrix'		
!RowID	!ColumnID	!Value
..	..	..

!!SBtab TableID='sm_example_2' TableType='SparseMatrixOrdered'		
!RowNumber	!ColummNumber	!Value
..	..	..

!!SBtab TableID='sm_example_3' TableType='SparseMatrixRow'		
!RowID	!RowString	
..	..	..

where RowString contains JSON strings describing all elements in the row and

!!SBtab TableID='sm_example_4' TableType='SparseMatrixColumn'		
!ColumnID	!ColumnString	
..	..	..

where ColumnString contains JSON strings describing all elements in the column.

3.7 Table for tool options

Table type Config This table type is used to specify tool options.

!!SBtab TableID='cf_example_1' TableType=Config' Method='parameter-balancing'	
!Option	!Value
ph	7
temperature	298.15
cell_volume	43

The option values (column Value) may be numbers, Boolean, or strings, as required by the respective tool. We recommend to give structured data as JSON strings. The table attribute Method is used to point to the software or calculation method by which the options are used. At the moment, it is used only by the parameter balancing tool [9].

4 SBtab conventions for metabolic state data

The SBtab format can be used to represent kinetic metabolic models and corresponding data (experimentally measured, or obtained from simulations). Relevant data include kinetic and thermodynamic constants as well as state-specific "omics" data, including metabolite concentrations, metabolic fluxes, reaction Gibbs free energies, enzyme concentrations, or mRNA abundances. The formats for model structures (including kinetic rate laws and kinetic and thermodynamic constants) are already described in the SBtab specification. This section describes, in addition, simple format conventions for "omics" data. This format is recommended for applications such as Parameter Balancing, Enzyme Cost Minimisation, Flux Cost Minimisation, Elasticity Sampling, or Convex Model Balancing.

4.1 Tables for a single metabolic state

To describe a single metabolic state, we can use the table type 'Quantity' and the same format that we would also use kinetic constants. For metabolite concentrations, for example, we can write:

!!SBtab TableType='Quantity' SBtabVersion='1.2'				
!QuantityType	!Compound	!Value	!Unit	!Compound:Identifiers:kegg.compound
concentration	NAD	1.6	mM	C00003
concentration	ATP	2	mM	C00002
...				

In the example, the entries in the columns QuantityType and Unit are always the same. Such columns can be omitted, and the entries can be given as table attributes instead: QuantityType='concentration' Unit='mM'. Therefore, a compact version of the table above reads (with the SBtabVersion='1.2' argument omitted for brevity):

!!SBtab TableType='Quantity' QuantityType='concentration' Unit='mM'		
!Compound	!Value	!Compound:Identifiers:kegg.compound
NAD	1.6	C00003
ATP	2	C00002
...		

For data with mean value und standard deviation, instead of a single value, we can write:

!!SBtab TableType='Quantity' QuantityType='concentration' Unit='mM'			
!Compound	!Mean	!Std	!Compound:Identifiers:kegg.compound
NAD	1.6	0.1	C00003
ATP	2	0.2	C00002
...			

Remarks:

- **Geometric mean values and standard deviation** For data with geometric mean values and standard deviation, we can use columns GeomMean and GeomStd instead.
- **Reaction column** A table for kinetic constants would contain columns that refer to reactions. Since these columns are not necessary in this case, they can be omitted.
- **Annotations** The table can contain metabolite names and / or metabolite annotations. However, users are highly encouraged to provide at least one common annotation (in the case of metabolites, for example, KEGG, ChEBI, or BIGG identifiers).
- **Order of columns** The order of columns is arbitrary as always
- **SBtab document** Like all SBtab tables, the data table can be added to a model in an SBtab document (possibly, containing a model and other state data tables). As always, SBtab identifiers must be unique within the SBtab document.

Comparable tables for metabolic fluxes, reaction Gibbs free energies, and enzyme levels read

!!SBtab TableType='Quantity' QuantityType='flux' Unit='mM/s'		
!Reaction	!Value	!Reaction:Identifiers:kegg.reaction
HXK_R01786	14.9	R01786
PGI_R02740	12.2	R02740
...		

!!SBtab TableType='Quantity' QuantityType='Gibbs free energy of reaction' Unit='kJ/mol'		
!Reaction	!Value	!Reaction:Identifiers:kegg.reaction
HXK_R01786	10	R01786
PGI_R02740	8	R02740
...		

!!SBtab TableType='Quantity' QuantityType='concentration of enzyme' Unit='mM'		
!Gene	!Value	!Gene:LocusName
Crp	1872	b3357
Ndk	19029	b2518
...		

Of course, the types of names and annotations used in these examples are just suggestions. Generally, each biological element (compound, reaction, etc) must be unique denoted by a name, a general annotation, or an SBtab or SBML ID referring to a model element. For reactions, alternatively, a reaction formula may be given.

Recommended annotations for different types of biological elements are as follows:

Metabolites	KEGG	Identifiers:kegg.compound
	ChEBI	Identifiers:chebi
	BIGG	Identifiers:bigg.metabolite
Reactions	KEGG	Identifiers:kegg.reaction
	BIGG	Identifiers:bigg.reaction
Enzymes	EC	Identifiers:ec-code
	Uniprot	Identifiers:uniprot
Proteins	Uniprot	Identifiers:uniprot

4.2 Tables for multiple metabolic states

For data matrices (e.g. an $m \times n$ metabolomics data table for m metabolites and n metabolic states), the SBtab specification contains different alternative recommendations. Here we propose another one, which is a bit simpler. We use the table type 'QuantityMatrix' (as proposed in the specification). The data columns are uncontrolled, i.e., without "!" in the header. Although column headers can be chosen at will, as rules of good practice we recommend the following naming. Columns are called like the samples they describe:

!!SBtab TableType='QuantityMatrix' QuantityType='concentration' Unit='mM'				
!Compound	Aerobic glucose	Anaerobic glucose	Acetate	!Compound:Identifiers:kegg.compound
NAD	1.6	0.1	1.6	C00003
ATP	2	0.2	2	C00002
...				

To specify mean values and standard deviations, we use headers of the form [SAMPLE]:Mean and [SAMPLE]:Std, for example

!!SBtab TableType='QuantityMatrix' QuantityType='concentration' Unit='mM'					
!Compound	Glucose:Mean	Glucose:Std	Acetate:Mean	Acetate:Std	!Compound:Identifiers:kegg.compound
NAD	1.6	0.1	1.6	0.1	C00003
ATP	2	0.2	2	0.2	C00002
...					

Tables for other quantities (for example fluxes, reaction Gibbs free energies, enzyme amounts) can be written in the same way.

5 SBtab tables for sparse matrices and relations

SBtab defines table types for describing pairwise, and possibly quantitative, relationships. The table type 'Relation' (with columns !From and !To, as well as optimal columns !Relation and !Value) describes relationships between SBtab elements (IDs given in columns From and To), where the type of relationship can be declared by the table attribute or in a column. For example, the table attribute Relation=is_a may be replaced by a table column !Relation, with entries `is_a`. Multiple relationships can be defined in a single table (optional column Relation, containing different strings denoting the different relations). Quantitative values can be given in an optional column Value.

Quantitative relationships can also be expressed by matrices. For convenience, SBtab provides four different formats for sparse matrices.

- TableType = 'SparseMatrix' (with columns !RowID, !ColumnID, !Value)
- TableType = 'SparseMatrixOrdered' (with columns !RowNumber !ColumnNumber, !Value)
- TableType = 'SparseMatrixRow (with columns !RowID and !RowString, where RowString is a JSON string describing all elements in the row)
- TableType = 'SparseMatrixColumn (with columns !ColumnID and !ColumnString, where ColumnString is a JSON string describing all elements in the column)

Example: an ontology In computer science, an ontology is a set of elements together with a set of relations between these elements. In SBtab, this can be represented by a table for elements and their properties, and a table (of type Relation) for one or several relations. Here is an example (with elements stemming from an invented Molecule table, and with a quantitative **has_part** relation):

!!SBtab TableType='Molecule' TableID='Molecule'	
!ID	!Name
A	a
B	b
...	...

!!SBtab TableType='Relation' TableID='Relation'			
!From	!To	!Type	!Value
A	B	is_a	
C	B	is_a	
B	X	has_part	2
..	..	..	..

SBtab documents and relational databases SBtab documents can be converted into relational databases. In this interpretation, SBtab tables can describe types of objects or relations between objects.

- **Tables for objects** To describe a type of objects, an SBtab table must contain an !ID column, containing SBtab identifiers serving as unique keys. The other columns describe properties of the object.
- **Tables for relations** Binary relations are described by tables of the table type Relation. Each row contains the IDs of two SBtab elements (in columns !From and !To), as well as the type of relation and a value.

Comments: when converting an SBtab document into a relational database, all tables must either contain an !ID column, or have the TableType Relation (and contain !From and !To columns). Moreover, all table columns must be controlled (i.e., start with ! or >). In the conversion, all other columns will be ignored.

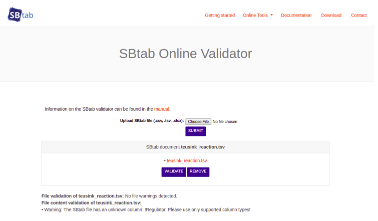
6 Software tools

Working with SBtab is facilitated by a number of online and offline tools (see `www.sbtab.net`). The SBML $\leftrightarrow$ SBtab converter relies on SBML-specific table and element definitions (as defined by the default

Definitions file) and on our reaction formula syntax. All other tools work independently of the type of content of the SBtab document.

6.1 SBtab online tools

1. Online validator for SBtab files. The online validator tool checks whether SBtab files (in .tsv, .csv, or .xlsx format) adhere to the SBtab conventions introduced in this manuscript. If a problem is identified by the validator, an instruction on how to fix the problem is provided. The validation is based on the SBtab table definitions found in the Definition table.



Information on the SBtab validator can be found in the manual.

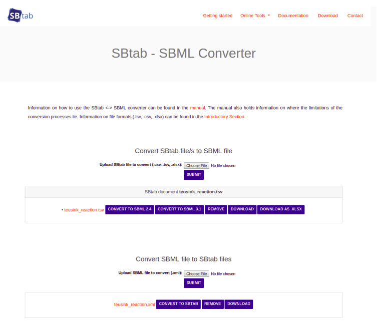
Upload SBtab file (.csv, .tsv, .xlsx):

SBtab document **teusink_reaction.tsv**

- teusink_reaction.tsv

File validation of teusink_reaction.tsv: No file warnings detected.
File content validation of teusink_reaction.tsv:
- Warning: The SBtab file has an unknown column: !Regulator. Please use only supported column types!

2. Online SBtab ↔ SBML converter The online conversion tool can create SBtab files from SBML models and vice versa. For the conversion from SBtab to SBML, it has to be assured that at least an SBtab table of type Reaction or Compound is provided. As additional information, the following SBtab table types can be used for the conversion to SBML: Compartment, Quantity, Gene, FbcObjective, and Layout. All information comprised in these SBtab tables can be converted to the SBML structure, as long as they are adhering to the correct syntax. Therefore, it is recommended to validate the SBtab files with the online validator before recruiting them for a conversion to SBML. The generated SBtab files can be displayed online as HTML tables. If annotations are correctly provided, they will link to the web resource. For the conversion, it is recommended to use SBML Level 2, Version 4, or higher. The table types FbcObjective and Layout are specific to SBML Level 3 packages and can only be employed for SBML models Level 3, Version 1 or higher. The details on the conversions can be read in Chapter 7.1. Note that structured Miriam annotations (column MiriamAnnotation) are not yet supported.



Information on how to use the SBtab <-> SBML converter can be found in the manual. The manual also holds information on where the limitations of the conversion processes lie. Information on file formats (.tsv, .csv, .xlsx) can be found in the Introductory Section.

Convert SBtab file/s to SBML file

Upload SBtab file to convert (.csv, .tsv, .xlsx):

SBtab document **teusink_reaction.tsv**

- teusink_reaction.tsv

Convert SBML file to SBtab files

Upload SBML file to convert (.xml):

teusink_reaction.xml

6.2 Python code

Python code The SBtab software tools have been programmed in Python3. The source code[6] can be employed via pip packages or as Linux commandline tool. To install the pip package, please type `sudo pip3 install pbalancing` in your Linux commandline. Then, you can import the pbalancing module in your Python3 projects. To use the Python3 code via the commandline, simply clone the Github project and follow the instructions on the Github project page. At `github.com/tlubitz/SBtab/python/api_documentation` you will find an extensive programming API for the Python modules as well as more detailed instructions on the installation and usage of the Python files.

Pandas data frames SBtab supports the usage of pandas data structures. The SBtab method `SBtab.to_data_frame()` exports an SBtab object to a pandas dataframe. The method `SBtab.from_data_frame()` imports a pandas dataframe to an SBtab object. For more information on the API please see the documentation at `github.com/tlubitz/SBtab/python/api_documentation`.

6.3 Visualisation in HTML

Using python, an SBtab table or document can be converted into HTML via the commandline tool `sbtab_sbtab2html.py`. For more information, see the documentation at `github.com/tlubitz/SBtab/`

[6]`github.com/tlubitz/SBtab`

python/api_documentation. The tool is a wrapper around the python method `misc.sbtab_to_html()`. if you use this method directly, it is important to set the parameter `mode='standalone'`.

6.4 Code and tools for R and Matlab

Interface to SBtab python code in the R software environment The R project is a widespread and powerful software framework for statistical computing and graphics. We provide a simple interface that allows R users to import the SBtab python code into their R projects, manipulate SBtab objects, call Python functions from within the R environment. For this, we use the rPython library, which needs to be preinstalled to employ the provided code. The R files, which can be found in the SBtab repository on Github, exemplify how this can be done. SBtab objects can be created via the Python interface:

- SBtab tables can be validated and the output stored as R variable
- SBtab tables can be edited (adding/removing rows, attributes, or columns) from within R
- SBtab tables can be converted to SBML and vice versa, where the output can be stored as R file objects

Alongside these rPython-based approaches, we also translated the generation of an SBtab file document (including one or more SBtabs) from Python directly into the R language. These example files are stored in the SBtab repository (/SBtab/R/).

Matlab Code for importing and exporting SBtab files in Matlab is available at the SBtab website. The Matlab toolbox handles only tab-delimited text files and does not support the .xls format.

7 Modelling tools and online resources

7.1 Conversion to and from SBML

SBML (Systems Biology Markup Language) models can be converted into SBtab documents and, conversely, SBtab documents describing models can be converted into SBML. Depending on the content of the SBML model, the SBtab files can comprise table types Reaction, Compound, Compartment, and Quantity. Likewise, these SBtab table types can be converted into an SBML (Level 2, Version 4 or Level 3, Version 1) model. The conversion to SBML, however, requires at least either a Reaction or Compound table, since this is the minimal information for creating an SBML model.

SBML $\rightarrow$ SBtab The conversion from an SBML model file to SBtab translates the structural and temporal information of the model into corresponding SBtab table files. The (i) Reaction SBtab contains a list of the reactions of the SBML file, including their reaction sum formula, kinetic laws, irreversibility, annotations, and more. Note that the SBML modifiers of a reaction (e.g. enzymes) cannot be identified as inhibitor or stimulator if they are not assigned an SBO Term within the SBML code. If this is not the case, they will only be exported to SBtab as modifiers without regulatory information. All species from the model can be found in the (ii) Compound SBtab. Their location, charge, annotations, and more are provided in the SBtab. Analogously, a (iii) Compartment table holds all structural information of the cellular compartments. The (iv) Quantity SBtab file lists all local and global parameters that are part of the model. Also their numerical values and units will be provided. The parameters can appear as either local or global variables in the SBML code; this information will be transferred to SBtab as well.

SBtab $\rightarrow$ SBML In the conversion from SBtab to SBML, Compound entries in SBtab correspond to `species` elements in SBML. By default, the unique keys (from the !ID column in the Compound and Reaction tables) are used as `id` attributes of the SBML elements. If SBML IDs are directly specified within SBtab (in the columns SBML:reaction:id, SBML:species:id, SBML:parameter:id, SBML:reaction:parameter:id, etc), these will be used instead. Rate laws from the SBML code are stored in SBtab as strings within a KineticLaw column. The syntax corresponds to the syntax of kinetic rate laws defined in the SBML specification. Note that the rate laws are not checked for their validity. It is up to the user to assure the correctness of the rate laws. If they are erroneous, this leads to invalid SBML output. An automatic parser of rate laws including checks of validity is planned for future versions of SBtab. Regulator entries in SBtab correspond to `modifier` elements in SBML; multiple regulators can be described by a regulation formula (in the Regulator column): regulators are separated by a "|" symbol, while the sign of regulation can be denoted by + or -. For an enzyme allosterically inhibited by ATP and activated by ADP, the formula reads `-ATP|+ATP` or `ATP|ADP` where inhibition and activition remain unspecified. Also rate rules and assignment rules can be converted from SBtab to SBML. Note that, just like for kinetic rate laws, these rules do not underlie constraints of validity. It is up to the user to ensure their correctness before conversion to SBML. For all aforementioned SBML elements, annotations are automatically translated from the SBtab to the SBML file, if they adhere to the correct syntax.

Model parameters The entries of Quantity tables can be inserted into SBML models or be extracted from them. By default, SBtab quantities referring to a reaction will become local reaction parameters in SBML, while other quantities become global parameters. The element ID of the !ID column will be used as SBML element ID unless it is overridden by the (optional) column !SBML:parameter:id (for global parameters) or !SBML:reaction:parameter:id (for local reaction parameters). Naming conventions for kinetic constants are given in [6], supplementary material Table A.5. Quantities that describe initial species amounts, initial species concentrations, or compartment sizes will be translated into the corresponding SBML element attributes.

SBML features not supported by SBtab There are some limitations to the conversion of SBtab and SBML. So far, the conversion does not include element notes, function definitions, rules, and events, as well support for the other SBML Level 3 packages. We plan to support these features in future versions of SBtab.

SBML Level 3 core The SBML Level 3, Version 1 format requires the declaration of certain attributes for model entities, which were optional in the previous SBML versions. If these attributes are not set in an SBML model, the code is invalid. To avoid this, the SBML converter will set these attributes to default values if no values are found in the SBtab file/s. The mandatory attributes are (default values set in brackets) as follows. Species elements require `ID` (!ID column), `Compartment` (default compartment), `HasOnlySubstanceUnits` (False), `boundaryCondition` (column !BoundaryCondition, False), and `constant` (column !IsConstant, False). Reaction elements require `ID` (!ID column), `reversible` (column !IsReversible, True), and `fast` (False). For species reference elements the mandatory attributes are `ID` (!ID column) and `constant` (True). Compartment elements also require `ID` (!ID column) and `constant` (!IsConstant column, True). Finally, parameters require `ID` (!ID column), `constant` (True), `value` (!Value column), and `units` (!Unit column).

SBML Level 3 packages SBML Level 3 comprises a number of additional packages for special purposes. SBtab supports the SBtab $\leftrightarrow$ SBML conversion of two of these packages:

- **fbc package** SBtab supports the FBC package for storing flux balance constraints in constraint-based (i.e. steady-state) models. The SBtab table type FbcObjective (new in SBtab version 1.2) stores a part of the required information and the other part is directly added to the table types Reaction, Compound, and Gene. For this as well, you can find example files on `www.sbtab.net` and the corresponding specification in Appendix A.
- **layout package** The Layout package is used for storing the spatial topology of a network diagram[7]. This type of information is stored in the SBtab table type Layout (new in SBtab version 1.2). For an example, see section 3.4. More example files can be found on our website `www.sbtab.net` and the allowed columns for the table type are listed in Appendix A of this specification.

SBtab columns referring to SBML Level 3 packages are represented by a specific column syntax, i.e. !SBML:*packagename*:*attribute*, where *packagename* corresponds to either layout or fbc and *attribute* is the name of the SBML attribute of the package. As an example, the upper bound of a reaction flux in a constraint-based model is given in a column !SBML:fbc:upperBound in a Reaction SBtab. This syntax rule holds for all columns *outside* the table types Layout and FbcObjective, which are solely employed for the usage in the corresponding Level 3 packages and do not need a more explicit label.

7.2 Modelling tools supporting SBtab

Parameter balancing Parameter balancing is a computational method for obtaining consistent parameter sets for kinetic metabolic models. Balanced parameter sets, computed from measured kinetic constants and other experimental data, respect constraints between biochemical quantities and assumptions about typical ranges, represented by prior values and bounds. A tool for parameter balancing [9] supporting SBtab is available at `www.parameterbalancing.net`.

Equilibrator The eQuilibrator is a tool for computing reaction thermodynamic parameters [10] based on the component contribution method [11]. The online version at `equilibrator.weizmann.ac.il` provides a convenient interface to thermodynamics calculations, as well as predictions of optimal metabolic states using the MDF [12] and ECM [13] method.

[7]Following the convention in SBML, all size information given for layout objects are understood to be in Point (pt), which is defined to be 1/72 of an inch (0.3527777778 mm) as in postscript. The origin of the coordinate system will be in the upper left corner of the screen. The positive x-axis runs from left to right, the positive y-axis run from top to bottom and the positive z-axis points into the screen.

References

[1] M. Kanehisa, S. Goto, S. Kawashima S, and A. Nakaya. The KEGG databases at genomenet. *Nucleic Acids Research*, 30:42–46, 2002.

[2] C. Laibe and N. Le Novère. MIRIAM Resources: tools to generate and resolve robust cross-references in Systems Biology. *BMC Systems Biology*, 1(1):58, 2007.

[3] M. Hucka, A. Finney, H.M. Sauro, H. Bolouri, J.C. Doyle, H. Kitano, A.P. Arkin, B.J. Bornstein, D. Bray, A. Cornish-Bowden, A.A. Cuellar, S. Dronov, E.D. Gilles, M. Ginkel, V. Gor, I.I. Goryanin, W.J. Hedley, T.J. Hodgman, J.H. Hofmeyr, P.J. Hunter, N.S. Juty, J.L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L.M. Loew, D. Lucio, P. Mendes, E. Minch, E.D. Mjolsness, Y. Nakayama, M.R. Nelson, P.F. Nielsen, T. Sakurada T J.C. Schaff, B.E. Shapiro, T.S. Shimizu, H.D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and the SBML Forum. The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[4] M. Courtot et al. Controlled vocabularies and semantics in systems biology. *Molecular Systems Biology*, 7(543), 2011.

[5] N. Juty, N. Le Novère, and C. Laibe. Identifiers.org and MIRIAM Registry: community resources to provide persistent identification. *Nucleic Acids Research*, 40:D580–D586, 2012.

[6] W. Liebermeister, J. Uhlendorf, and E. Klipp. Modular rate laws for enzymatic reactions: thermodynamics, elasticities, and implementation. *Bioinformatics*, 26(12):1528–1534, 2010.

[7] N. Le Novère, A. Finney, M. Hucka, U.S. Bhalla, F. Campagne, J. Collado-Vides, E.J. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B. Shapiro, J.L. Snoep, H.D. Spence, and B.L. Wanner. Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat Biotech.*, 23(12):1509–1515, Dec 2005.

[8] `http://www.ebi.ac.uk/compneur-srv/miriam-main/`.

[9] T. Lubitz and W. Liebermeister. Parameter balancing: consistent parameter sets for kinetic metabolic models. *Bioinformatics*, 35:3857–3858, 2019.

[10] A. Flamholz, E. Noor, A. Bar-Even, and R. Milo. eQuilibrator – the biochemical thermodynamics calculator. *Nucleic Acids Research*, 40(D1):D770–D775, 2012.

[11] E. Noor, H.S. Haraldsdottir, R. Milo, and R.M.T. Fleming. Consistent estimation of Gibbs energy using component contributions. *PLoS Computational Biology*, 9:e1003098, 2013.

[12] E. Noor, A. Bar-Even, A. Flamholz, E. Reznik, W. Liebermeister, and R. Milo. Pathway thermodynamics highlights kinetic obstacles in central metabolism. *PLoS Computational Biology*, 10:e100348, 2014.

[13] E. Noor, A. Flamholz, A. Bar-Even, D. Davidi, R. Milo, and W. Liebermeister. The protein cost of metabolic fluxes: prediction from enzymatic rate laws and cost minimization. *PLoS Computational Biology*, 12(10):e1005167, 2016.

[14] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.

Acknowledgements

The authors thank Dagmar Waltemath, Hans-Michael Kaltenbach, Dirk Wiesenthal, Jannis Uhlendorf, Anne Goelzer, Jörg Büscher, Avi Flamholz, Elad Noor, Edda Klipp, Frank Bergmann, Phillipp Schmidt, Matthias König, Oliver Bodeit, and Jonathan Karr for their contributions. This work was funded by the German Research Foundation (LI 1676/2-1).

A Overview of the SBtab format

A.1 Summary of SBtab rules

We summarise the most important conventions implemented by SBtab:

- **SBtab identifiers** Document or table attribute names, column names, and element IDs (entries in ID columns) should respect the rules for SBtab identifiers: must start with a letter and must not contain whitespaces or the : or . characters. Element IDs should be unique within an SBtab document.
- **Element IDs** SBtab elements (e.g. describing compounds) are referred to by element IDs, which are defined in the corresponding table (e.g. Compound for compounds). The (non-mandatory) column !ID contains the element IDs, which serve as primary keys for this table. Element IDs must start with a letter and must not contain whitespaces or the special characters ":" or ".". They must be unique within an SBtab document.
- **Column order** The allowed column types depend on the table type, but their order is arbitrary. The only exception is the first column !ID, which contains the element IDs (acting as keys for this table). However, it is good practice to sort the columns by importance and to arrange related columns next to each other (e.g. placing a column Value next to a column Unit).
- **ASCII Characters** The table fields contain only plain text. The format is case-sensitive, but the choice of fonts (bold, italic) does not play a role. Double quotes should not be used within cells.
- **Decimal points** Float numbers must use decimal points (instead of decimal commas).
- **Table types and column names** Table types and their possible columns are defined in appendix A. Column names may not contain any special characters or whitespaces (parsers should ignore these characters).
- **Comment lines** Table lines starting with a "%" character contain comments and are ignored during parsing.
- **Comments and references** Additional information about table elements can be stored in the optional columns !Comment, !Reference, !Reference:Identifiers:pubmed, and !ReferenceDOI, which can appear in all tables.
- **Unrecognised table or columns** Columns with unknown headers (not starting with !), or unrecognised header starting with ! may appear in SBtab tables. They can be used, but are not supported by the parser. The use of undefined columns is inadviseable.
- **Attribute line** The first line, starting with !!SBtab must declare at least the attributes: TableType, TableName, and possibly the properties SBtabVersion (for SBtab version used) and Document (the element ID of the SBtab document that contains the table). The entries must be separated by whitespaces.
- **Identifiers** Identifiers for compounds, compartments etc. can be specified in columns with a header "*ElementType*:Identifiers:*DB*").
- **Missing elements** If an element is missing, the table field is left empty. Missing numerical values can also be indicated by non-numerical elements like `?` or `na` (for "not available"). Mandatory fields must not be empty.
- **Formulae** Reaction sum formulae and kinetic rate laws must be written in the format described in section 3.1.
- **Reserved names** In the SBtab format, there are reserved names for (i) table types (marked by colours in this text); (ii) column names; (iii) types of biological elements (see Table 26); and (iv) types of biochemical quantities or mathematical terms (e.g. Mean) for them (see Table 27), and physical units.
- **Physical units** In SBtab, it is recommended to use the units listed in the SBML specification (see sbml.org/Documents/Specifications and section B). As good practice, derived units (e.g. kJ/mol) and reciprocal units (e.g. 1/s) should be given in the simplest possible form, in necessary using multiplication, division, exponentials, and round brackets (e.g. gram/m^3).

Use of special characters Here is a summary of special characters in SBtab and their recommended use: The colon : is used in database column headers in between the actual column header and the database name: ..ID:db.

- Round brackets `()` are used as brackets in mathematical formulae.
- Square brackets `[]` are used for function arguments in quantitative formulae, around the biological role in biological role formulae, for compartments in names of localised entities, and for physical units.
- The pipe symbol `|` is used in lists of alternative names, IDs, etc and in Boolean formulae (as or).
- Single or multiple exclamation point symbols `!` are used to mark extra rows in tables of type OmicsDataRow or OmicsDataColumn and within column headers that refer to rows of other tables (example in OmicsDataRow tables referring to row of a QuantOLD table).
- The character “&” is used in Boolean formulae (and).

Use of special characters in formulae or expressions

- The symbols `+`, `-`, `*`, `/` are used in quantitative formulae.
- The symbols `&` and `|` are used in Boolean formulae.
- The symbols `+`, `-`, and `+-` are used in simple regulation formulae.
- The symbols `+` and `<=>` are used in stoichiometric sum formulae.

A.2 General document and table attributes

Document attributes

Name	Type	Format	Content
DocumentID	element ID	string	Document element ID
DocumentName	text	string	Document title (free text)
SBtabVersion	text	string	SBtab version number
Date	text	string	Date in YYYY-MM-DD
ReferenceDescription	text	string	Name of reference description
ReferenceCitation	text	string	Citation, unique identifier, unambiguous URL
ModelCreators	text	string	Name and contact information for model creators
ModelCreationTime	text	string	Date and time of model creation and last modification
TermsOfDistribution	text	string	Terms of distribution

Table 2: Document attributes, which can appear in the attribute line. The attributes in the lower part would be necessary for MIRIAM compliance. If ReferenceCitation contains a Pubmed Id, the attribute ReferenceCitation:Identifiers:pubmed should be used instead. ReferenceCitation should also identify the author/s of the model.

Table attributes

Name	Type	Format	Mandatory	Content
TableID	element ID	string	✓	Table element ID
TableType	text	string	✓	Table type (as defined in definition table)
TableName	text	string		Table name
Document	text	string		SBtab document name

Table 3: Table attributes (to appear in attribute line). Note that existing document attributes (e.g. Date) should not be repeated in individual tables.

A.3 General table columns

All table types

Name	Type	Format	Content
!Description	text	string	Description of the row element
!Comment	text	string	Comment
!ReferenceName	text	string	Reference title, authors, etc. (as free text)
!Reference:Identifiers:pubmed	text	string	Reference PubMed ID
!ReferenceDOI	text	string	Reference DOI

Table 4: Columns that can appear in all tables. These columns are listed here for convenience; in the Definitions.tsv table, they must be declared individually for each table type.

All entity and reaction tables

Name	Type	Format	Content
!ID	text	string	Element element ID
!Name	text	string	Entity name
!Identifiers:*DataCollection*	resource ID	string	Entity ID
!MiriamAnnotation	annotation	string	Entity ID (JSON string)
!Type	text	string	Biochemical type of entity (examples see Table 26)
!Symbol	text	string	Short symbol (e.g., gene symbol)

Table 5: Columns that can appear in all entity (i.e. Compound, Enzyme, Gene, Regulator, and Compartment) and Reaction tables. These columns are listed here for convenience; in the Definitions.tsv table, they are declared individually for each table type.

Definition

Name	Type	Content
!ComponentName	component name	Name of component (table, column, attribute to be defined)
!ComponentType	`Table, Column, Attribute`	Type of component
!IsPartOf	component name	name of parent component
!Format	String	Format
!linksElementID	string	Boolean flag indicating if component links to element ID of a component from other table
!Description	Text	Free text description of component

Table 6: Columns that can appear in Definition tables.

A.4 Table types in the SBtab default format

Compound

Name	Type	Format	Content
!ID	element ID	string	Compound element ID
!SBML:species:id	SBML element ID	string	SBML Species ID of the entity
!SBML:speciestype:id	SBML element ID	string	SBML SpeciesType ID of the entity
!InitialValue	number	float	Initial amount or concentration
!Unit	string	string	Unit for initial value
!Location	element ID	string	Compartment for localised entities
!State	element ID	string	State of the entity
!CompoundSumFormula	text	string	Chemical sum formula
!StructureFormula	text	string	Chemical structure formula
!Charge	number	integer	Electrical charge number
!Mass	number	float	Molecular mass
!Unit	text	string	Physical unit
!IsConstant	Boolean	Boolean	Substance with fixed concentrations
!HasOnlySubstanceUnit	Boolean	Boolean	Numbers represent amounts (not concentrations)
!BoundaryCondition	Boolean	Boolean	Dynamics not fully determined by reactions
!EnzymeRole	element ID	string	Enzymatic activity
!RegulatorRole	element ID	string	Regulatory activity
!SBML:fbc:ChemicalFormula	text	string	Chemical formula of the species
!SBML:fbc:Charge	number	float	Charge of the SBML species

Table 7: Columns that can appear in Compound tables

Enzyme

Name	Type	Format	Content
!ID	element ID	string	Enzyme element ID
!CatalysedReaction	element ID	string	Catalysed reaction
!KineticLaw:Name	name	string	Rate law (name as in SBO)
!KineticLaw:Identifiers.obo.sbo	element ID	string	Rate law SBO identifier
!KineticLaw:Formula	string	string	Rate law formula
!Pathway	text	string	Pathway name (free text)
!Gene	element ID	string	Gene coding for enzyme (element ID)
!Gene:Name	string	string	Gene coding for enzyme (name)
!Gene:Symbol	string	string	Gene coding for enzyme (short symbol)

Table 8: Columns that can appear in Enzyme tables

Protein

Name	Type	Format	Content
!ID	element ID	string	Protein element ID
!Name	text	string	Protein name
!Symbol	string	string	Protein symbol
!Gene	element ID	string	Gene element ID
!Gene:Name	text	string	Gene name
!Gene:Symbol	string	string	Gene symbol
!Gene:LocusName	string	string	Gene locus name
!Mass	number	number	Protein mass
!Size	number	number	Protein chain length

Table 9: Columns that can appear in Protein tables

Gene

Name	Type	Format	Content
!ID	element ID	string	Gene element ID
!Name	text	string	Gene name
!Symbol	string	string	Gene symbol
!LocusName	string	string	Gene locus name
!GeneProduct	element ID	string	Gene product element ID
!GeneProduct:Name	string	string	Gene product name
!GeneProduct:Symbol	string	string	Gene product symbol
!GeneProduct:SBML:species:id	SBML element ID	string	SBML ID of protein
!Operon	element ID	string	Operon in which gene is located
!SBML:fbc:ID	element ID	string	Gene ID in the package
!SBML:fbc:Name	text	string	Gene name in package
!SBML:fbc:GeneProduct	Boolean	string	Boolean flag indicating if the gene is an FBC gene product
!SBML:fbc:GeneAssociation	Boolean	string	Boolean flag indicating if the gene is an FBC gene association
!SBML:fbc:Label	text	string	Gene label in the package

Table 10: Columns that can appear in Gene tables

Regulator

Name	Type	Format	Content
!ID	element ID	string	Regulator element ID
!State	element ID	string	State of the regulator
!TargetGene	element ID	string	Target gene
!TargetOperon	element ID	string	Target operon
!TargetPromoter	element ID	string	Target promoter

Table 11: Columns that can appear in Regulator tables

Compartment

Name	Type	Format	Content
!ID	element ID	string	Compartment element ID
!Identifiers:obo.sbo	element ID	string	Compartment SBO term
!SBML:compartment:id	SBML element ID	string	SBML Compartment ID
!OuterCompartment	element ID	string	Surrounding compartment (short)
!OuterCompartment:Name	string	string	Surrounding compartment (name)
!OuterCompartment:SBML:compartment:id	SBML element ID	string	Surrounding compartment
!Size	number	float	Compartment size
!Unit	text	string	Physical unit
!IsConstant	Boolean	Boolean	Indicates a constant compartment size

Table 12: Columns that can appear in Compartment tables

Reaction

Name	Type	Format	Content
!ID	element ID	string	Reaction element ID
!SBML:reaction:id	SBML element ID	string	SBML Reaction ID
!ReactionFormula	ReactionFormula formula	string	Reaction sum formula
!Location	element ID	string	Compartment for localised reaction
!Enzyme	element ID	string	Enzyme catalysing the reaction
!Regulator	element ID	string	Regulator controlling the reaction
!Model	text	string	Model(s) in which reaction is involved
!Pathway	text	string	Pathway(s) in which reaction is involved
!SubreactionOf	element ID	string	Mark as subreaction of a (lumped) reaction
!IsComplete	Boolean	Boolean	Reaction formula includes all cofactors etc.
!IsReversible	Boolean	Boolean	Reaction should be treated as irreversible
!IsInEquilibrium	Boolean	Boolean	Reaction approximately in equilibrium
!IsExchangeReaction	Boolean	Boolean	Some reactants are left out
!IsFast	Boolean	Boolean	Reaction to be treated as fast
!Flux	number	float	Metabolic flux through the reaction
!IsNonEnzymatic	Boolean	Boolean	Non-catalysed reaction
!KineticLaw:Name	name	string	Rate law (name as in SBO)
!KineticLaw:Identifiers.obo.sbo	element ID	string	Rate law SBO identifier
!KineticLaw:Formula	string	string	Rate law formula
!Gene	element ID	string	see table type Enzyme
!Gene:Symbol	string	string	see table type Enzyme
!Operon	element ID	string	see table type Gene
!Enzyme:Name	string	string	Name of enzyme
!Enzyme:Identifiers:ec-code	string	string	EC number of enzyme
!Enzyme:SBML:species:id	SBML element ID	string	SBML ID of enzyme
!Enzyme:SBML:parameter:id	SBML element ID	string	SBML ID of enzyme
!Enzyme:SBML:reaction:parameter:id	SBML element ID	string	SBML ID of enzyme
!BuildReaction	Boolean	Boolean	Includereaction in SBML model
!BuildEnzyme	Boolean	Boolean	Include enzyme in SBML model
!BuildEnzymeProduction	Boolean	Boolean	Describe enzyme production in SBML model
!SBML:fbc:GeneAssociation	Boolean	string	Gene association in the FBC package
!SBML:fbc:LowerBound	number	float	Lower bound of reaction flux
!SBML:fbc:UpperBound	number	float	Upper bound of reaction flux

Table 13: Columns that can appear in Reaction tables. The lower section lists, again, column types from Table A.4.

ReactionStoichiometry

Name	Type	Format	Content
!Reaction	element ID	string	Reaction ID
!Stoichiometry	number	number	Stoichiometric coefficient
!Substrate	element ID	string	Substrate ID
!Product	element ID	string	Product ID
!Location	element ID	string	Compartment ID

Table 14: Columns that can appear in ReactionStoichiometry tables.

FbcObjective

Name	Type	Content
!ID	string	Element ID of the FBC objective
!Name	string	Name of the FBC objective
!Type	string	Type of the FBC objective
!Active	Boolean	Flag indicating if the objective is active
!Objective	string	FBC objective as sum of coefficients times reaction IDs

Table 15: Columns that can appear in FbcObjective tables.

Position

Name	Type	Format	Content
!Element	element ID	string	Element displayed
!PositionX	number	float	x coordinate for graphical display
!PositionY	number	float	y coordinate for graphical display
!Label	string	string	Name to be displayed in graphics

Table 16: Columns that can appear in Position tables

Layout

Name	Type	Content
!ID	string	Unique identifier of layout package
!Name	string	Name of layout in plugin
!ModelEntity	string	Declares the SBML model entity
!SBML:compartment:id	string	SBML identifier of the compartment
!SBML:reaction:id	string	SBML identifier of the reaction
!SBML:species:id	string	SBML identifier of the species
!CurveSegment	string	Type of curve segment (Start, End, BasePoints)
!SBML:X	float	Coordinate on X axis
!SBML:Y	float	Coordinate on Y axis
!SBML:width	float	Width of the element
!SBML:height	float	Height of the element
!SBML:text	string	Text label of the element
!SBML:speciesRole	string	Role of the species in the reaction (Reactant, Product)

Table 17: Columns that can appear in Layout tables.

Relation

Name	Type	Format	Content
!Relation	string	string	Type of quantitative relationship
!From	element ID	string	Element at beginning of arrow
!To	element ID	string	Element at arrowhead
!IsSymmetric	Boolean	Boolean	Flag indicating non-symmetric relationships
!Value:*QuantityType*	number	float	Numerical value assigned to the relationship

Table 18: Columns that can appear in Relation tables.

Config

Name	Type	Format	Content
!Option	text	string	Tool option name
!Value	text	string	Tool option value

Table 19: Columns that can appear in Config tables.

Quantity

Name	Type	Format	Content
!ID	element ID	string	Quantity / SBML parameter ID
!QuantityName	string	string	Quantity (name)
!QuantityType	element ID	string	Quantity type (e.g. from SBO)
!Value	number	float	Simple value
!Mean	number	float	Algebraic mean
!Std	number	float	Standard deviation (positive)
!Min	number	float	Lower bound
!Max	number	float	Upper bound
!Median	number	float	Median
!GeometricMean	number	float	Geometric mean
!Sign	sign	string	Sign
!ProbDist	text	string	Probability distribution
!SBML:parameter:id	SBML element ID	string	Parameter ID in SBML file
!SBML:reaction:parameter:id	SBML element ID	string	Parameter ID in SBML file
!Unit	text	string	Physical unit
!Scale	text	string	Scale (e.g. logarithm, see Table 24)
!IsConstant	Boolean	Boolean	Indicates a constant quantity
!Time	number	float	Time value
!Sample	string	string	Sample name or identifier
!Provenance	text	string	Name of data source (free text)
!Condition	text	string	experimental condition name (free text)
!pH	number	float	pH value in measurement
!Temperature	number	float	Temperature in measurement
!Location	element ID	string	Compartment (element ID)
!Location:Name	string	string	Compartment (name)
!Location:SBML:compartment:id	SBML element ID	string	SBML ID of compartment‘
!Compound	element ID	string	Related compound (element ID)
!Compound:Name	string	string	Related compound (name)
!Compound:Identifiers:*DataCollection*	resource ID	string	Compound ID
!Compound:SBML:species:id	SBML element ID	string	SBML ID of compound
!Reaction	element ID	string	Related reaction (element ID)
!Reaction:Name	string	string	Related reaction (name)
!Reaction:Identifiers:*DataCollection*	resource ID	string	Reaction ID
!Reaction:SBML:reaction:id	SBML element ID	string	SBML ID of reaction
!Enzyme	element ID	string	Related enzyme (element ID)
!Enzyme:Name	string	string	Related enzyme (name)
!Enzyme:Identifiers:*DataCollection*	resource ID	string	Enzyme ID
!Enzyme:SBML:species:id	SBML element ID	string	SBML ID of enzyme
!Enzyme:SBML:parameter:id	SBML element ID	string	SBML ID of enzyme
!Enzyme:SBML:reaction:parameter:id	SBML element ID	string	SBML ID of enzyme
!Protein	element ID	string	Related enzyme (element ID)
!Protein:Name	string	string	Related enzyme (name)
!Protein:Identifiers:*DataCollection*	resource ID	string	Protein ID
!Protein:SBML:species:id	SBML element ID	string	SBML ID of enzyme
!Protein:SBML:parameter:id	SBML element ID	string	SBML ID of enzyme
!Protein:SBML:reaction:parameter:id	SBML element ID	string	SBML ID of enzyme
!Gene	element ID	string	Related gene
!Organism	element ID	string	Related organism

Table 20: Columns for numerical values and experimental conditions in tables of type Quantity.

QuantityMatrix

Name	Type	Format	Content
!Time	number	float	Time value
!Sample	string	string	Sample name or identifier
!¿ *Table*:*Column*	pointer	string	Pointer to column in anther table
!¿ *Document*:*Table*:*Column*	pointer	string	Pointer to column in anther document
!Quantity	string	string	Quantity (element ID)
!QuantityName	string	string	Quantity (name)
!QuantityType	element ID	string	Quantity type (e.g. from SBO)
ValueType	ValueType	string	Mathematical Term from table 24
!SBML:parameter:id	SBML element ID	string	Parameter ID in SBML file
!SBML:reaction:parameter:id	SBML element ID	string	Parameter ID in SBML file
!Unit	text	string	Physical unit
!Scale	text	string	Scale (e.g. logarithm, see Table 24)
!Compound	element ID	string	Related compound (element ID)
!Compound:Name	string	string	Related compound (name)
!Compound:Identifiers:*DataCollection*	resource ID	string	Compound ID
!Compound:SBML:species:id	SBML element ID	string	SBML ID of compound
!Reaction	element ID	string	Related reaction (element ID)
!Reaction:Name	string	string	Related reaction (name)
!Reaction:Identifiers:*DataCollection*	resource ID	string	Reaction ID
!Reaction:SBML:reaction:id	SBML element ID	string	SBML ID of reaction
!Enzyme	element ID	string	Related enzyme (element ID)
!Enzyme:Name	string	string	Related enzyme (name)
!Enzyme:Identifiers:*DataCollection*	resource ID	string	Enzyme ID
!Enzyme:SBML:species:id	SBML element ID	string	SBML ID of enzyme
!Enzyme:SBML:parameter:id	SBML element ID	string	SBML ID of enzyme
!Enzyme:SBML:reaction:parameter:id	SBML element ID	string	SBML ID of enzyme
!Protein	element ID	string	Related enzyme (element ID)
!Protein:Name	string	string	Related enzyme (name)
!Protein:Identifiers:*DataCollection*	resource ID	string	Protein ID
!Protein:SBML:species:id	SBML element ID	string	SBML ID of enzyme
!Protein:SBML:parameter:id	SBML element ID	string	SBML ID of enzyme
!Protein:SBML:reaction:parameter:id	SBML element ID	string	SBML ID of enzyme
!Gene	element ID	string	Related gene

Table 21: Columns that can appear in QuantityMatrix tables.

QuantityInfo

Name	Type	Format	Content
QuantityType	string	string	QuantityType (e.g. from SBO)
Symbol	string	string	Symbol for the quantity type
Unit	string	string	Unit for the Median value
Constant	string	string	Is this value constant
Element	string	string	A related element
RelatedElement	string	string	A related element
Scaling	string	string	A scaling factor
Dependence	string	string	Is this value dependent on others
PriorMedian	float	float	Prior median value
PriorStd	float	float	Prior median standard deviation
LowerBound	float	float	Lower bound for this quantity type
UpperBound	float	float	Upper bound for this quantity type
ErrorStd	float	float	Error of standard deviation
DataStd	float	float	Error of standard deviation
SBMLElement	string	string	Corresponding SBML element
SBMLElementType	string	string	Corresponding SBML element
Abbreviation	string	string	Abbreviation of quantity type
ID	string	string	Abbreviation of quantity type
MatrixInfo	string	string	Description of parameter dependencies

Table 22: Columns for numerical values and experimental conditions in tables of type QuantityInfo used in parameter balancing.

Measurement

Name	Type	Format	Content
!Sample	element ID	string	Sample element ID
!Time	number	integer or float	Time point
!Unit	Unit	string	Measurement unit
!ValueType	ValueType element	string	Mean, Std, etc
!Description	text	string	Free text description of sample

Table 23: Columns that can appear in Measurement tables.

B Predefined terms and recommended controlled vocabularies

Physical units As a rule of good practice, SBtab files should use the units supported by SBML[8]. Orders of magnitude can be denoted by `k, M, c, m, mu, n, p, f` for Kilo, Mega, Centi, Milli, Micro, Nano, Pico, Femto. If a parameter is dimensionless, it has to be annotated as `dimensionless`. Arbitrary units should be denoted by `arbitrary unit` (note that this does not exist in SBML).

Unknown values and Boolean values Unknown data values are written as **NaN**. Boolean values are written as **True** and **False** (unless otherwise specified by a software).

ValueType	Type	Format	Meaning
Value	number	float	Simple value
Mean	number	float	Algebraic mean
Std	number	float (positive)	Standard deviation
Min	number	float	Lower bound
Max	number	float	Upper bound
Median	number	float	Median
GeometricMean	number	float	Geometric mean
GeometricStd	number	float	Geometric standard deviation
Sign	sign	{+,-,0}	Sign
ProbDist	Free text	string	Prob. distribution

Scale	Meaning
Lin	Linear scale
Ln	Natural logarithm
Log2	Dual logarithm
Log10	Decadic logarithm

Table 24: Terms for mathematical quantities and mathematical scales recommended for use in SBtab. Names of probability distributions can be, for instance, `Normal, Uniform, LogNormal`.

Database	MIRIAM URN	Contents	URI
SBO	obo.sbo	Quantities, rate laws	`www.ebi.ac.uk/sbo/`
CheBI	obo.chebi	Metabolites	`www.ebi.ac.uk/chebi/`
Enzyme nomenclature	ec-code	Enzymes	`www.ebi.ac.uk/IntEnz/`
KEGG Compound	kegg.compound	Compounds	`www.genome.jp/KEGG/`
KEGG Reaction	kegg.reaction	Reactions	`www.genome.jp/KEGG/`
KEGG Orthology	kegg.orthology	Genes	`www.genome.jp/KEGG/`
UniProt	uniprot	Proteins	`www.uniprot.org/`
SGD	sgd	Yeast gene loci	`www.yeastgenome.org/`
Gene Ontology	obo.go	Compartments	`www.geneontology.org/`
Taxonomy	taxonomy	Organisms	`www.ncbi.nlm.nih.gov/Taxonomy/`
SGD	sgd	Yeast proteins	`www.yeastgenome.org/`

Table 25: A selection of recomended databases. For a complete list, see the MIRIAM resources [2].

Physical entity types	
protein complex	SBO:0000297
messenger RNA	SBO:0000278
ribonucleic acid	SBO:0000250
deoxyribonucleic acid	SBO:0000251
polypeptide chain	SBO:0000252
polysaccharide	SBO:0000249
metabolite	SBO:0000299
macromolecular complex	SBO:0000296

Compartments	
cell	GO:0005623
extracellular space	GO:0005615
membrane	GO:0001602
cytosol	GO:0005829
nucleus	GO:0005634
mitochondrion	GO:0005739

Table 26: Examples of biochemical entity types (with Systems Biology Ontology identifiers [4]) and cell compartments (with Gene Ontology identifiers [14]).

[8]`ampere, gram, katal, metre, second, watt, becquerel, gray, kelvin, mole, siemens, weber, candela, henry, kilogram, newton, sievert, coulomb, hertz, Litre, ohm, steradian, dimensionless, item, lumen, pascal, tesla, farad, joule, Lux, radian`

Name	SBO term	Default unit	Entities
standard Gibbs energy of formation	SBO:0000582	kJ/mol	Compound
standard Gibbs energy of reaction	SBO:0000583	kJ/mol	Compound
equilibrium constant	SBO:0000281	variable	Reaction
forward maximal velocity	SBO:0000324	mMol/s	Enzymatic reaction
reverse maximal velocity	SBO:0000325	mMol/s	Enzymatic reaction
substrate catalytic rate constant	SBO:0000321	1/s	Enzymatic reaction
product catalytic rate constant	SBO:0000320	1/s	Enzymatic reaction
Michaelis constant	SBO:0000027	mM	Enzyme, Compound
inhibitory constant	SBO:0000261	mM	Enzyme, Compound
activition constant	SBO:0000363	mM	Enzyme, Compound
Hill constant	SBO:0000190	dimensionless	Compound, Reaction
concentration	SBO:0000196	mM	Compound
biochemical potential	SBO:0000303	kJ/mol	Compound
standard biochemical potential	SBO:0000463	kJ/mol	Compound
amount of an entity pool	SBO:0000361)	–	Compound, Protein
mass of an entity pool	SBO:0000504)	–	Compound, Protein
rate of reaction	SBO:0000612	–	Reaction
rate of reaction (amount)	SBO:0000615	mol/s	Reaction
rate of reaction (concentration)	SBO:0000614	mM/s	Reaction
Gibbs free energy of reaction	SBO:0000617	kJ/mol	Reaction
standard Gibbs free energy of formation	SBO:0000582	kJ/mol	Compound
standard Gibbs free energy of reaction	SBO:0000583	kJ/mol	Compound
transformed standard Gibbs free energy of reaction	SBO:0000620	kJ/mol	Reaction
transformed standard Gibbs free energy of formation	SBO:0000621	kJ/mol	Compound
transformed Gibbs free energy of reaction	SBO:0000622	kJ/mol	Reaction
thermodynamic temperature	SBO:0000147	K	Location (optional)
ionic strength	SBO:0000623	mM	Location (optional)
pH	SBO:0000304	dimensionless	Location (optional)
molecular mass	SBO:0000647	Dalton	Compound
protein molecular mass	SBO:0000648	Dalton	Protein or Enzyme
protein chain length	SBO:0000667	dimensionless	Protein
yield	SBO:0000668	–	–
biomass yield on substrate	SBO:0000669	–	–
product yield on substrate	SBO:0000670	–	–

Table 27: A selection of quantity types to be used in SBtab in table types Quantity. The unit of equilibrium constants depends on the reaction stoichiometry. More quantities can be found in the Systems Biology Ontology [4].

C Changes since SBtab version 1.0

Format and conventions

- Support for SBML3 fbc package: table type FbcObjective
- Support for SBML3 layout package: table type Layout
- The table attribute TableID is mandatory, while TableName is a free string. The attribute TableTitle is deprecated.
- Document and table attributes must appear in the first cell of the table attribute line, separated by whitespaces. In its present version, the parser still tolerates other quotation marks and multiple whitespaces or tabs as separators.
- The values of document and table attributes must be given in simple quotation marks 'XX' (UTF-8 code 27). Some other quotation marks are tolerated by the parser, but their use is not recommended.
- Underscores are allowed in IDs.
- Recommended use of JSON (e.g. for MIRIAM annotations)
- Document attribute line
- ID column is not mandatory; good practice: put ID column first in a table.
- linksShortname column in Definitions table
- Decimal points (instead of decimal commas) are mandatory

Implementation

- Implementation in python3
- Python version with unit tests and pip installer
- Improved conversion to and from pandas dataframes
- Improved conversion to HTML
- SBML3 mandatory attributes
- The excel format .xlsx is supported, while .xls is not supported anymore
- The file extension .tab is not recommended anymore; instead, .tsv is recommended