SBtab version 2.0 Specification

Conventions for structured data tables in Systems Biology

Wolfram Liebermeister[1], Timo Lubitz[2], and Jens Hahn[2]

[1] Institut für Biochemie, Charité - Universitätsmedizin Berlin
[2] Institut für Biophysik, Humboldt-Universität zu Berlin

Abstract

Data tables in the form of spreadsheets or delimited text files are the most utilised data format in Systems Biology. However, they 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, shortnames, 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 found at www.sbtab.net.

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

Contents

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 2.0.

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 [**?**]. 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 shortnames 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='1.0'
!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 have special meanings (the colours themselves are just used in this text and are not part of the SBtab format). The SBtab table differs from the original table in several ways: the first line (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 mandatory TableID must be provided, but can be chosen arbitrarily. However, it should adhere to some syntax conventions explained later. The version 1.0 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[1]. The syntax of the reaction formulae is also uniquely defined. In particular, the shortnames 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 shortnames can be defined by providing standardised names or database identifiers in a second table of type Compound. The compound shortnames will then serve as keys to rows of this table.

[1]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 [**?**]) is guaranteed to remain stable in time.

!!SBtab SBtabVersion='1.0'	TableID='compound_example'	TableType='Compound'
!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 practise, they can be stored as separate files, as sheets of a spreadsheet file, or within a single table. The following example contains all necessary information to build a stoichiometric model in the SBML (Systems Biology Markup Language) format [**?**] (a software tool for conversion between SBtab and SBML is described in section 4):

!!SBtab SBtabVersion='1.0'	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
% That's a comment			
!!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
...	...	...	...

Here, 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 shortnames do not comply with SBML's rules for element identifiers. Also note the line starting with the % character: it's a comment line that is ignored by the parser.

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 an other optional attribute to the first line, TableName, which can be chosen freely and does not underlie 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) [**?**]. 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 [**?**]. In this case, the table would read:

!!SBtab	TableID='quantity_example_2'	TableType='Quantity'			
!ID	!QuantityType	!SBML:reaction:id	!SBML:species:id	!Value	!Unit
MyData_1	equilibrium constant	r1		0.156	dimensionless
MyData_2	Michaelis constant	r1	atp	0.96	mM
MyData_3	inhibition constant	r1	atp	0.13	mM
MyData_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 [**?**]) 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'		
!CompoundID	!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.

In the following sections, we introduce the general SBtab rules (specification for SBtab version 2.0), as well as formats and conventions for different types of use (see Section 2). It We defines a list of table types (see Section 3) and explains the syntax of reaction formulae in the SBtab format (see Section 2.5). Finally, the specification references the available online tools for the handling of SBtab files (see Section 5) and includes an overview of all available SBtab table types in appendix C. Appendix D lists controlled vocabularies and database resources recommended to be used within SBtab.

2 Overview of the SBtab format

2.1 Basic conventions

SBtab comprises a list of conventions about the structure, nomenclature, syntax, and annotations in tables describing biochemical network models, kinetic parameters, and dynamic data. It contains

1. General rules for the **structure of tables** and the **syntax** used in table fields.
2. Defined **table types** for different kinds of information, each with possible **columns** with defined names and data types (see Table 1; An overview of all predefined table types and their possible columns is given in the appendix).
3. A **syntax for biochemical element annotations** pointing to databases or ontologies.
4. Rules for usage of **names**, **shortnames**, and **database identifiers** used for annotation.
5. **Naming rules for biochemical quantities** to specify the quantities, physical units, and mathematical terms (like `Mean` for mean values).
6. A syntax for **reaction sum formulae**.
7. A mechanism for **extending the format** by declaring new column or table types.

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. This is reflected by the table types defined in Table 1.

Colour highlighting and predefined terms In the examples shown in this text, predefined SBtab entries are highlighted in colours. This is just for convenience and is not a part of the SBtab format. `Table types` and `Column types` defined by the SBtab format are listed in Table 1. `Shortnames` can be chosen *ad hoc* by the user; each of them needs to be defined by a table row. Shortnames 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.

2.2 SBtab tables and SBtab documents

General table structure An SBtab document consists of one or several tables that refer to a common model or related data sets. All tables `within the document` must use a common list of shortnames. For instance, a `Compound` table contains the column `!ID`, and the elements from this column define compound shortnames to be used in the other tables. Several tables in a document may have the same type, but their table `identifiersnames` (indicated by the table attribute `TableNameID`) must be unique within the document.

Table declaration row and table attributes The top left field contains the table header, starting with `!!SBtab` and followed by the table attributes in the syntax *attribute name='attribute value'*. Different quotation marks (including " or ') may be used. However, we recommend to use ' as single quotation mark for the attribute value (and the double quote " to enclose cells in comma-delimited table files). `Different attributes can either be placed in separate table cells, or all be placed in the first cell and separated by whitespaces.` One `mandatory attribute is TableID, which must be provided as unique identifier of the table within the document. It can be chosen arbitrarily, but must not contain special characters or whitespaces. It may not start on a digit. If 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. `The other mandatory An optional` table attribute is `TableName`, defining `an arbitrary name for the table, which does not underlie syntax restrictions. the table's shortname (respecting the syntax for shortnames, i.e., no whitespaces or special characters, not starting with a numerical character). The optional attribute TableTitle`

can contain a free-text title. Other recommended table attributes are liste in table 3. Any other table attributes can be added, as long as their names are valid shortnames (no whitespaces or special characters, not starting with a numerical character). The very name of a table attribute can refer to an SBtab element. 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. A Date attribute can be declared as well.

Document declaration row and document attributes SBtab documents consist of one or more SBtab tables. A document can begin with an optional document declaration row, which starts with !!!SBtab. Analogous to aforementioned table declaration line, the document declaration line holds the defining attributes. Most importantly, it requires an attribute Document, which is an identifier via which the document can be referred to (e.g. in the Document attribute in table declaration lines, which defines to which document a table belongs). Optionally, a DocumentName and DocumentType can be declared, as well as the Date.

Header row, column names, and definition table The second row 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 good practise is to set ID as the first column. A column with identical entries for all rows can be replaced, for simplicity by a table attribute starting with ':': for instance, a row with header !Unit and entries 'min' can be replaced by the table attribute :Unit='min'.

Column with unique keys Unique keys for table rows can be defined in a column ID. The keys must be unique across the entire SBtab document and must satisfy the syntax rules for shortnames. It is good practise to make the first column the ID column.

Completeness To interpret the contents of a single table, other tables (e.g. describing shortnames) may be required. If a table does not require any other tables, we call it "complete". A document is complete if all names are defined, i.e. no unspecified information is required to interpret its contents. If a single table or a document are incomplete, the undefined names have to be known by the software, and an exchange with other software tools is likely to fail. If a table or document contains two elements, and there is no explicit information implying that they describe the same things, it is assumed that they describe different things.

Conventions for spreadsheet files To ensure consistency between 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 , or .tab), a document can either be stored in several files with the filenames *basename_tablename.extension*, or tables are concatenated vertically, each preceded by a declaration row (starting with !!), and stored in a single table file. In the latter case, the document may be preceded by a document declaration line !!!SBtab....
- **Delimiters in .tsv or .csv, or .tab files** In character-delimited files, it is recommended to use the filename extension ".tsv" (or ''.tab'') 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 quotation marks ("..").
- Excel data files SBtab files do not necessarily have to be character-delimited, but may also be tables stored as Microsoft Excel documents with the .xlsx-extension.

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	FBC objective information	model configuration
Layout	Layout graph information	model layout
Measurement	Measurements or samples	quantitative data
Quantity	Individual data for model parameters	quantitative data
QuantityMatrix	Data matrices	quantitative data
QuantityInfo	Information on quantities	informative
PbConfig	Configuration file for parameter balancing tool	software configuration
Relation	Graphs and relations	network data
Definition	Define custom column types, etc.	customising SBtab

Table 1: Overview of table types predefined in SBtab.

JSON columns SBtab supports the usage of JSON syntax within the table columns. This is practical for the provision of more than one biochemical annotation (see Chapter 2.4 for more details). The allowed syntax for a JSON column in SBtab reads {"A":"X", "B":"X"}. Make sure to employ ASCII supported quotation marks.

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).

2.3 Names of biochemical elements

Names and identifiers of model elements In the following, compounds, enzymes, genes, genetic regulators, and compartments will be called "biochemical entities". "Biochemical elements" comprises, in addition, reactions and biochemical quantities. Biochemical elements can be described by shortnames, official names, or database identifiers (IDs). The shortnames have to be declared within the SBtab document and have to satisfy syntactic rules. Each table starts with a column ID of the same name, containing the shortnames. Shortnames, the arbitrary element names used in a data set or model, must be unique, i.e. declared only once in a document; they must start with a letter and may not contain spaces or the special characters ":", ".". 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 22). Sometimes, elements may be characterised redundantly: e.g. the reaction catalysed by an enzyme, given in an Enzyme table, can be given by both shortname and database ID. In case of conflict, the information derived from the shortname (i.e. the database ID listed in the Reaction table) has higher priority.

Naming and specification of biological entities Tables of the types Compound, Enzyme, Gene, Regulator, or Compartment are called "entity tables". The biochemical meaning of the entities 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 [**?**] (data collection and ID).

Localised compounds If a compound, enzyme, or genetic regulator is localised in a compartment, the corresponding localised entity can be denoted by *compound*[*compartment*] with square brackets, where *compound* and *compartment* are the shortnames or IDs of the compound and the compartment used in the model. If a model contains several compartments, tools should treat the first compartment in the Compartment table as the standard compartment. The standard compartment will be used by default for all compounds that are not explicitly assigned to compartments.

2.4 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[2]. 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. 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"]]
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".

[2]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.

2.5 Syntax for 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 shortnames, 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. The localisation in compartments can be denoted as follows:

- Reaction in the default compartment: `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 shortnames, and `comp1` and `comp2` are compartment shortnames. The reversibility of reactions is not encoded in the reaction sum formula, but defined by an extra column `!IsReversible` in the `Reaction` table.

3 Overview of predefined table types

SBtab predefines a number of table types with specific properties. An overview is given in Table 1. The table types `Compound`, `Enzyme`, `Gene`, `Regulator`, `Compartment`, and `Reaction` describe model structures, the table types `Quantity`, `QuantityMatrix`, and `Relation` are used for quantitative data.

3.1 Tables for biochemical network structures

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 shortname 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 C.2.
- **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 12. 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).

3.2 The table type `ReactionStoichiometry` for biochemical network structures

The following example shows how a `ReactionStoichiometry` table can be used to describe the stoichiometric structure of a reaction network. 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.

3.3 Table type Quantity 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' shortnames. There are two 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 11).

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 shortnames 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

3.4 Table types QuantityMatrix and Measurement 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='quantity_example_5'	TableType='Quantity'	TableName='Concentration'
!ID	!Identifiers:obo.chebi	!QuantityType	!Unit
Glucose	CHEBI:17234	concentration	mM
Fructose	CHEBI:15824	concentration	mM

Now let us consider data tables in which time points are represented by columns. A similar scheme can be used in this case. The first, simple version would read:

!!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

Here, 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 shortnames 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		

3.5 The 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

3.6 Table type Definition for customising the SBtab format

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. 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_example'	TableType='Definition'			
!ComponentName	!ComponentType	!IsPartOf	!linksShortname	!Format	!Description
SBML:reaction:id	Column	Reaction	True	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, Shortname (name of SBtab element, as defined in one of the SBtab tables), Number (integer or float in usual formats, or complex numbers like 1 + i 3), or Boolean (with possible values True and False, or alternatively 1 and 0). More specific string formats (e.g., for reaction sum formulae) are currently not formally defined, but can be mentioned in the Description column. The linksShortname column states whether the component may link to a shortname of a component from another table in form of a Boolean variable.

3.7 Sparse matrices in the SBtab format

To be written: SparseMatrixOrdered, SparseMatrix, SparseMatrixRow, SparseMatrixColumn.

4 Conversion between SBtab and 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 SBtab, as these are the minimal information for creating an SBML model.

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. (v) Events can be an important part of SBML models; they indicate e.g. concentration changes or stress applications at certain time points. They too are translated into an SBtab file. Finally, (vi) rules are exported from SBML to an SBtab Rule table. Rules can comprise assignment rules, algebraic rules, and rate rules. Rule formulas and units are part of the conversion as well.

In the conversion from SBtab to SBML, Compound entries in SBtab correspond to species elements in SBML. By default, the unique keys of the !ID column in the Compound and Reaction SBtab 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. Finally, SBtab is able to provide lists of events for the SBML file. This includes the event assignments, triggers, delays, and more. For all aforementioned SBML elements, annotations are automatically translated from the SBtab to the SBML file, if they adhere to the correct syntax.

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 shortname 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 [**?**], 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.

4.1 SBML Level 3

SBML (Level 3, Version 1) models require the declaration of certain attributes for model

entities, which were optional in previous SBML versions. If these attributes are not set, the SBML code can be considered invalid or at least erroneous. Thus, our SBML converter sets these attributes to default values, if they are not provided by the user in the SBtab file/s. The mandatory attributes are (default values set in brackets): for species elements we require ID (!ID column), Compartment (default compartment), HasOnlySubstanceUnits (False), boundaryCondition (False), and constant (False). For reaction elements we require ID (!ID column), reversible (True), and fast (False). For species reference elements the attributes are ID (!ID column) and constant (True). Compartment elements also require ID (!ID column) and constant (True). Finally, parameters require ID (!ID column), constant (True), value (!Value column), and units (!Unit column).

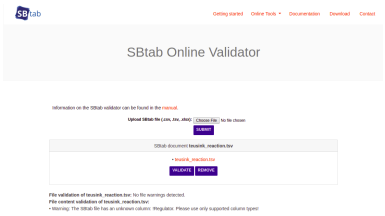
Furthermore, SBML Level 3 comprises a number of additional packages for special purposes. SBtab supports the bidirectional conversion of two of these packages: The Layout package is employed for storing the spatial topology of a network diagram. This type of information is stored in the SBtab table type Layout (new in SBtab version 2.0). Example files are on our website www.sbtab.net and the allowed columns for the table type are in Appendix C of this specification. 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. Also, SBtab supports the FBC package. which is employed for storing flux balance constraints in constraint-based (a.k.a. steady-state) models. The SBtab table type FbcObjective (new in SBtab version 2.0) 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 C.

All columns related 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:upperBou in a Reaction SBtab. This syntax rule is 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.

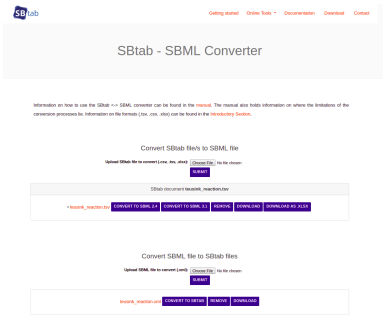
There are still limitations to the conversion of SBtab and SBML. So far, the conversion does not include element notes and function definitions. These issues are planned to be solved in future versions of SBtab as well as the support of Events, Rules, and further SBML Level 3 packages.

5 SBtab tools

To simplify the usage of SBtab, we provide several online tools at www.sbtab.net.



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.



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`, `Events, and Rules` `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 for SBML Level 3 packages and can thus only be employed for SBML models Level 3, Version 1 or higher. The details on the conversions can be read in Chapter 4. Note that structured Miriam annotations (column `MiriamAnnotation`) are not yet supported.

5.1 Python Tools

The SBtab software tools are programmed in Python3 and the source code is provided on github.com/tlubitz/SBtab. The code can be employed via pip packages or as Linux commandline tool. For the pip package you simply need to type sudo pip3 install pbalancing on your Linux commandline. Then, you can import the pbalancing module in your Python3 projects. To use the Python3 code via commandline, simply clone the Github project and follow the instructions on the Github project page. Also, there is an extensive programming API for the Python modules on github.com/tlubitz/SBtab/python/api_documentation and more detailled instructions on the installation and usage of the Python files.

Python parser for SBtab files. In addition, we provide a SBtab parser written in Python. It uses the Python package tablib to import SBtab files and provides different functions for editing the data and for directly accessing them. These features are important for the embedding of the SBtab file parser into software projects. The common operations for manipulating SBtab files contain:

1. Extracting characteristic table information (type, name, etc.).
2. Addition of rows and columns to the SBtab table.
3. Editing and export of the table content in rows, columns, and single entries. An

export as a Python dictionary is also possible, to ensure easy access to the data for python programmers.

4. Switching of columns and rows in the table (matrix transposition). As some data are stored conveniently in transposed spreadsheets, some tables need to be transposed to have better access to its content.
5. Duplicate SBtab objects.
6. Translate between reaction formulae and stoichiometric matrix (see section 3.5).
7. Writing SBtab files to the hard disk.

Also pandas data structures are supported: the method toDataFrame() returns a pandas.DataFrame() object with the data that is in the SBtab, including the column names.

5.2 Other Tools

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_fra imports a pandas dataframe to an SBtab object. For more information on the API please see the documentation on github.com/tlubitz/SBtab/python/api_documentation.

SBtab HTML visualiser You can generate HTML content from an SBtab object via the method misc.sbtab_to_html(). It is important to provide the parameter mode='standalone'.

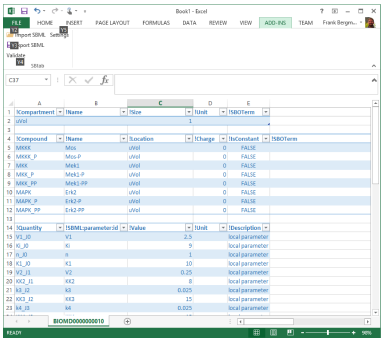
Simple interface for importing SBtab functionality to the R software environment Timo: "Wolf, is this interface still working? Most probably it will have to be adapted to the new API, right?" The R project is a widespread and powerful software framework for statistical computing and graphics. We provide 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/).

Support for MATLAB: Timo: "Wolf, is this interface still working? Most probably it will have to be adapted to the new API, right?"Code for importing and exporting SBtab files in MATLAB is available. The MATLAB toolbox expects SBtab files to be in tab-delimited text format and does not support .xls files.

3. MS Excel Add-in Timo: "Wolf, this interface is most probably not working anymore; should we ask Frank to fix it, take it out, or wait until we have the next publication of SBtab?" The described validator and converter functions can also be attained with an add-in for Microsoft Excel. It can be retrieved from the SBtab Github Repository and installed with a Windows Installer Package. The prerequisites for the installation of the add-in are (i) Windows Vista or higher, (ii) Microsoft Office 2010 or higher, (iii) Microsoft .NET Framework 4.5 (full) or higher, and Microsoft Visual Studio 2010 Tools for Office Runtime (VSTO). The latter two can be downloaded directly from Microsoft.



References

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, and Matthias König for contributing to this proposal. This work was funded by the German Research Foundation (LI 1676/2-1), the European Commission (projects BaSysBio and UNICELLSYS), and the German Federal Ministry of Education and Research (project OncoPath).

A Summary of SBtab rules

We summarise the most important conventions implemented by SBtab:

- **Shortnames** Model elements (e.g. compounds) are referred to by shortnames, which are defined in the corresponding table (e.g. `Compound` for compounds) . Shortnames must be unique within an SBtab document. The column `!ID` contains the shortnames, which serve as primary keys for this table and must therefore be unique.
- **Order of columns** 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 shortnames (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.
- **Decimal points** To simplify parsing, we recommend to use decimal points (instead of decimal commas).
- **Table types and column names** Table types and their possible columns are defined in appendix C. Column names may not contain any special characters or white spaces (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.
- **Declaration 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 shortname of the SBtab document that contains the table). The entries can be separated by whitespaces or be given in separate fields of the declaration line.
- **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 must be written in a special format explained below.
- **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 23); and (iv) types of biochemical quantities or mathematical terms (e.g. `Mean`) for them (see Table 24), and physical units.
- **Physical units** In SBtab, it is recommended to use the units listed in the SBML specification (see sbml.org/Documents/Specifications)[3]. 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`).

[3]The following units are supported by SBML: `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`. 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`.

B A note on MIRIAM-compliant models

The MIRIAM rules for computational models [**?**] 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 declaration line of the SBtab document in question, or in the declaration 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).

C Overview of table types

C.1 Document and table attributes and general column types

Document attributes

Name	Type	Format	Content
DocumentName	shortname	string	Document shortname
DocumentTitle	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 declaration row. 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	shortname	string	✓	Table shortname
TableType	text	string	✓	Table type (as defined in definition table)
TableName	shortname	string		Table name
TableTitle	text	string	✓	Table title (free text)
SBtabVersion	text	string		SBtab version number
Document	text	string		SBtab document name
Date	text	string		Date in YYYY-MM-DD

Table 3: Possible table attributes (to appear in declaration row).

C.2 Predefined table types

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

All entity and reaction tables

Name	Type	Format	Content
!ID	text	string	Element shortname
!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 23)
!Symbol	text	string	Short symbol (e.g., gene symbol)
!PositionX	number	float	x coordinate for graphical display
!PositionY	number	float	y coordinate for graphical display

Table 5: Columns that can appear in all entity (i.e. Compound, Enzyme, Gene, Regulator, and Compartment) and Reaction tables.

Compound

Name	Type	Format	Content
!ID	shortname	string	Compound shortname
!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	shortname	string	Compartment for localised entities
!State	shortname	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	shortname	string	Enzymatic activity
!RegulatorRole	shortname	string	Regulatory activity
!SBML:fbc:ChemicalFormula	text	string	Chemical formula of the species
!SBML:fbc:Charge	number	float	Charge of the SBML species

Table 6: Columns that can appear in Compound tables

Enzyme

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

Table 7: Columns that can appear in Enzyme tables

Protein

Name	Type	Format	Content
!ID	shortname	string	Protein shortname
!Name	text	string	Protein name
!Symbol	string	string	Protein symbol
!Gene	shortname	string	Gene shortname
!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 size

Table 8: Columns that can appear in Protein tables

Gene

Name	Type	Format	Content
!ID	shortname	string	Gene shortname
!Name	text	string	Gene name
!Symbol	string	string	Gene symbol
!LocusName	string	string	Gene locus name
!GeneProduct	shortname	string	Gene product shortname
!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	shortname	string	Operon in which gene is located
!SBML:fbc:ID	shortname	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 9: Columns that can appear in Gene tables

Regulator

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

Table 10: Columns that can appear in Regulator tables

Compartment

Name	Type	Format	Content
!ID	shortname	string	Compartment shortname
!Identifiers:obo.sbo	shortname	string	Compartment SBO term
!SBML:compartment:id	SBML element ID	string	SBML Compartment ID
!OuterCompartment	shortname	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 11: Columns that can appear in Compartment tables

`Reaction`

Name	Type	Format	Content
!ID	shortname	string	Reaction shortname
!SBML:reaction:id	SBML element ID	string	SBML Reaction ID
!ReactionFormula	ReactionFormula formula	string	Reaction sum formula
!Location	shortname	string	Compartment for localised reaction
!Enzyme	shortname	string	Enzyme catalysing the reaction
!Model	text	string	Model(s) in which reaction is involved
!Pathway	text	string	Pathway(s) in which reaction is involved
!SubreactionOf	shortname	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	shortname	string	Rate law SBO identifier
!KineticLaw:Formula	string	string	Rate law formula
!Gene	shortname	string	see table type `Enzyme`
!Gene:Symbol	string	string	see table type `Enzyme`
!Operon	shortname	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 12: Columns that can appear in `Reaction` tables. The lower section lists, again, column types from Table C.2.

`ReactionStoichiometry`

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

Table 13: Columns that can appear in `ReactionStoichiometry` tables.

`Relation`

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

Table 14: Columns that can appear in `Relation` tables.

`Quantity`

Name	Type	Format	Content
`!ID`	shortname	string	Quantity / SBML parameter ID
`!QuantityName`	string	string	Quantity (name)
`!QuantityType`	shortname	string	Quantity type (e.g. from SBO)
ValueType	ValueType	string	Mathematical Term from table 21
`!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 21)
`!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`	shortname	string	Compartment (shortname)
`!Location:Name`	string	string	Compartment (name)
`!Location:SBML:compartment:id`	SBML element ID	string	SBML ID of compartment‘
`!Compound`	shortname	string	Related compound (shortname)
`!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`	shortname	string	Related reaction (shortname)
`!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`	shortname	string	Related enzyme (shortname)
`!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`	shortname	string	Related enzyme (shortname)
`!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`	shortname	string	Related gene
`!Organism`	shortname	string	Related organism

Table 15: 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 (shortname)
`!QuantityName`	string	string	Quantity (name)
`!QuantityType`	shortname	string	Quantity type (e.g. from SBO)
ValueType	ValueType	string	Mathematical Term from table 21
`!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 21)
`!Compound`	shortname	string	Related compound (shortname)
`!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`	shortname	string	Related reaction (shortname)
`!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`	shortname	string	Related enzyme (shortname)
`!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`	shortname	string	Related enzyme (shortname)
`!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`	shortname	string	Related gene

Table 16: Columns that can appear in `QuantityMatrix` tables.

`Measurement`

Name	Type	Format	Content
`!Sample`	shortname	string	Sample shortname
`!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 17: Columns that can appear in `Measurement` tables.

`FbcObjective`

Name	Type	Content
`!ID`	`string`	`Shortname 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 18: `Columns that can appear in FbcObjective 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 19: Columns that can appear in Layout tables.

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
!linksShortname	string	Boolean flag indicating if component links to shortname of a component from other table
!Description	Text	Free text description of component

Table 20: Columns that can appear in Definition tables.

D Predefined terms and recommended controlled vocabularies

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
Sign	sign	{+,-,0}	Sign
ProbDist	Free text	string	Prob. distribution

Scale	Meaning
Lin	Linear scale (no transformation)
Ln	Natural logarithm
Log2	Dual logarithm
Log10	Decadic logarithm

Table 21: 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 22: A selection of databases to be used in SBtab. For the complete list, see the MIRIAM resources [**?**].

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 23: Examples of biochemical entity types (with Systems Biology Ontology identifiers [**?**]) and cell compartments (with Gene Ontology identifiers [**?**]).

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
rate of reaction (amount)	SBO:0000615	M/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)

Table 24: 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 [**?**].