Supporting information

EnzymeML - a data exchange format for biocatalysis and enzymology

Jan Range[1], Colin Halupczok[1], Jens Lohmann[1], Neil Swainston[2], Carsten Kettner[3], Frank T. Bergmann[4], Andreas Weidemann[5], Ulrike Wittig[5], Santiago Schnell[6], Jürgen Pleiss[1*]

[1] Institute of Biochemistry and Technical Biochemistry, University of Stuttgart, Allmandring 31, 70569 Stuttgart, Germany

[2] Institute of Systems, Molecular and Integrative Biology, University of Liverpool, Liverpool L69 7ZB, United Kingdom

[3] Beilstein-Institut, Trakehner Str. 7 – 9, 60487 Frankfurt am Main, Germany

[4] BioQUANT/COS, Heidelberg University, INF 267, Heidelberg, Germany

[5] Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnenweg 35, 69118 Heidelberg, Germany

[6] Department of Molecular & Integrative Physiology, and Department of Computational Medicine & Bioinformatics, University of Michigan Medical School, Ann Arbor, Michigan 48109, USA

[*] Corresponding author:
Jürgen Pleiss
Institute of Biochemistry and Technical Biochemistry
University of Stuttgart
Allmandring 31
70569 Stuttgart, Germany
E-mail: Juergen.Pleiss@itb.uni-stuttgart.de
ORCID: 0000-0003-1045-8202

1. Structure of EnzymeML documents

1.1 Content of the SBML file

1.1.1 `model` tag

In SBML, the `model` tag forms the container of all defined lists. The name of the experiment is saved in the `name` attribute. The metadata of the experiment file is saved as annotation in Dublin Core RDF format and contains information about the author of the file, the creation date, and the modification dates of the file. The information about the COMBINE Archive[1] and files included therein is saved as MIRIAM annotation. Information on the methods used in the different experiments is stored in the `reaction` tag.

1.1.2 Units

A strict definition of units is crucial to correctly describe the result of experiments and to ensure comparability and reproducibility. SBML defines SI units by the `listOfUnitDefinitions` tag, from which the experimentalist can choose. A new unit is defined by base units that can have a `scale`, a `multiplier`, and an `exponent`. Throughout the document, this unit is referred to by its `id`.

1.1.3 Compartments

The reactions take place in different environments, which are described in SBML with the `compartment` tag in the `listOfCompartments` and are annotated by MIRIAM. The `compartment` tag has attributes which describe whether it is `constant`, as well as its `size` and `units`. In EnzymeML, a compartment is a test tube, and each compartment describes a different experimental condition, such as initial substrate concentrations used in the reaction. Throughout the document, the compartment is referred to by its `id`, and its `name` describes the compartment in a human-readable form.

1.1.4 Species

In SBML, the `species` element defines an entity that is considered indistinguishable from each other, may participate in reactions, and is located in a specific compartment. The species element also defines attributes such as the `sboTerm` attribute (**Table S2**) which specifies the role of the species in the reaction, the `compartment` where the species is located, and the initial concentration or amount of the species. The latter can be described using the distribution package of SBML (*http://sbml.org/Documents/Specifications/SBML_Level_3/Packages/distrib*) to store value

distributions and ranges. The species name is stored in the `name` attribute, while further information about the species is stored in the respective annotations. The compartment description also includes other species such as buffer or additives, resulting in a comprehensive documentation of the reaction conditions.

1.1.5 Reaction

The reactions of the experiment are represented by the reaction element in the `listOfReactions` tag. They include the `listOfReactants`, the `listOfProducts`, and the `listOfModifiers` tags to represent the chemical equation of the experiment and their respective stoichiometry. Cascade reactions can be represented by different reaction elements in the same reaction vessel (defined via `compartment`) connected by the same species `id`s. Each reaction element is described by an `id`, `name`, and the information whether the reaction is `reversible`. The `listOfModifiers` tag gives information about the catalyst of the reaction and other species that interact with the reaction. The `kineticLaw` tag of SBML is used to describe a kinetic model fitted to the experimental data, including the mathematical model as well as parameters such as k_{cat}. The EnzymeML annotation `enzymeml:data` of the `listOfReactions` tag and the `reaction` tag specify the data stored in the CSV formatted experimental data file and links them to the respective `id`s of the species. The annotation `enzymeml:data` consists of three lists, beginning with `listOfFormats`, where columns for each CSV file found in the second list `listOfFiles` are described. Entries in `listOfMeasurements` denote which file belongs to a measurement which further links experiment metadata found in the EnzymeML document to the data.

1.1.6 Kinetic models

The `kineticLaw` tag describes the kinetic modelling process, where the estimated parameters are stored as `localParameters` and the model in MathML format, resulting in a comprehensive, reproducible description of the modelling procedure. The MIRIAM annotation provides further information about the model and the modelling method. Additional information can be stored as plain text. The EnzymeML annotation also provides information about the data used for the parameter estimation.

1.2 Experimental data file

The experimental data on substrate or product concentration as a function of time is stored in the tabular CSV format. Each column is separated by a comma, and each row is separated by a new line. This simple file format allows the user to easily write and read the file with a spreadsheet program such as Excel or a text editor. Because it is machine readable, it can also be generated and analyzed by a program.

The format of the experimental data file is described by the EnzymeML annotation in the experiment file and defines the content of the columns (time or concentration). Each concentration column refers to a species and a unit. Each experimental data file can contain multiple experiments, each experiment multiple replica and the `measurement id` and the `replica` attribute, respectively, are used to identify the begin and end row of a measurement or a replica.

1.3 Annotations

1.3.1 EnzymeML

EnzymeML annotations contain additional information that is not part of SBML or MIRIAM, such as pH value, temperature, and pressure (**Figure S1**). The annotations of a species element depend on its type. Small molecules are described by their IUPAC name,[2] their SMILES (simplified molecular-input line-entry system) code, [3] or their InChI (International Chemical Identifier) code,[4] proteins are described by their amino acid sequence. EnzymeML annotations in the `listOfReactions` (**Figure S2**) and `reaction` tags control the automated use of the data stored in the experimental data files, such as format, arrangement of data, and replica. This leads to a list of measurements, which can be later used for kinetic modelling. Furthermore, all datasets are assigned to their corresponding experiments, thus enabling reuse of data. A complete specification as XML Schema is available at GitHub (*https://github.com/EnzymeML/PyEnzyme/blob/main/xsd/EnzymeML.xsd*).

1.3.2 MIRIAM

The MIRIAM annotations[5] are stored in the RDF format (*http://www.w3.org/TR/rdf-schema/)* and structured in a subject-predicate-object format. The subject is usually the `metaid` attribute of the annotated element, while the predicate is defined using one of the BioModels Qualifiers (**Figure S3**). The object is an RDF Bag container with a list of elements, that include as a resource attribute a link to *http://identifiers.org* that links to the ontology resource. The output

format of the link can be requested as an RDF document, which is machine-readable and can be used by a program to retrieve further information about the annotated elements.

The RDF format is used to describe metadata. In the subject, which is represented by the `Description` element, the `metaid` is linked to the attribute `about` and the '#' sign followed by the `metaid`. The predicate determines how the object is represented by the subject. The subject in EnzymeML is any SBML element defined with `metaid`. The predicates are listed in the MIRIAM annotation or in other namespaces.

2. API and thin API layer

2.1 API

An application programming interface (API) which consists of two libraries, the Python library PyEnzyme (*https://github.com/EnzymeML/PyEnzyme/tree/main*, referred to below as <*GitHub*>) and the Java library JEnzyme (*https://github.com/EnzymeML/JEnzyme*), which support reading, writing, editing, merging, and visualization of EnzymeML documents. The Python library PyEnzyme can also be obtained via Python Packaging Index PyPI (*https://pypi.org/project/PyEnzyme*). The API uses the SBML syntax and naming conventions which are familiar to enzymologists to be implemented into EnzymeML. The basic concept of the two libraries is the usage of multiple dictionaries, in which proteins, reactants, units, and reactions are stored. They are indexed by internal IDs to prevent duplicates and ensure that they can always be traced back from reactions and *vice versa*. This allows an application to load multiple EnzymeML documents and filter them by user-defined properties, such as all reactions in which a certain reactant participates. EnzymeML objects, such as a protein, can be easily created and added to the respective dictionary by calling an `add`-function of an `EnzymeMLDocument` object. Similarly, an object can be retrieved by calling a `get`-function and its respective ID or name. All functions were optimised to require no further knowledge of either Java or Python, such that users only have to adapt to the EnzymeML syntax.

Since EnzymeML is a standardised data model, type checking and validation such as the range of allowed pH values is done consistently to maintain data quality. Prior to the creation of a reaction, all participating compounds have to be defined, otherwise an error message will indicate a possible inconsistency. Thus, data completeness is guaranteed and sparse EnzymeML documents are prevented.

The API also offers an export function of EnzymeML to any user-defined data format. Thus, large amounts of data included in multiple EnzymeML documents can be extracted and analyzed by machine learning methods purposes.

2.2 Application-specific thin API layer

In order to make EnzymeML accessible to a specific application, a thin API layer maps between the object layers of the API and of the application. An application-specific thin API layer can either be used to import or export EnzymeML documents. To create a thin API layer, two templates are provided which can be customised to read or write relevant attributes. The first template "TL_ImportTemplate.ipynb" (<*GitHub*>*/templates*) contains all code needed to extract information contained in an EnzymeML document. The user will decide which code is needed for the specific application and will do the mapping. The second template "TL_ExportTemplate.ipynb" (<*GitHub*>*/templates*) provides all code needed to create an EnzymeML document. The easy-to-use syntax allows fast and easy access to all information of an EnzymeML document as well as its creation. For instance, to extract a specific protein, the `EnzymeMLDocument` object inherits a get-function which either takes the internal ID or a given name as argument to return the respective object. The latter can then be used to extract and process all of its attributes. The export is done similarly: pre-defined objects are added via the `EnzymeMLDocument` objects add-function. It should be noted that the function also takes care of assigning internal IDs as well as parsing the unit string. Hence, users have to not to take care of technical aspects such as unit definitions and cryptical identifiers.

Three application-specific thin layers have been created to demonstrate the usage of the API for the integration of applications: TL_COPASI, TL_STRENDA, TL_BioCatNet (<*GitHub*>*/pyenzyme/Examples/ThinLayers*).

3. Applications

3.1 Creating EnzymeML documents from spreadsheets

Spreadsheets such as Excel files are widely used and serve as an easy way of storing data. Hence, we provide a thin API layer to generate an EnzymeML document from a spreadsheet provided by BioCatNet as a template. The data that was used describes the lyase-catalyzed self-ligation of 3,5-dimethoxybenzaldehyde[6]
(<*GitHub*>*/pyenzyme/Resources/Examples/ThinLayers/BioCatNet/DMBA_selfligation.xlsx*).

The thin API layer maps each object in the spreadsheet to an EnzymeML object such as

`Protein`, `Reactant`, or `Reaction` and adds it to the `EnzymeMLDocument` object by a simple `add`-function. The backend takes care of validating data types and provides programming language primitives to append information to the respective dictionary. After successful mapping and validation, the API object layer is then written to an EnzymeML document by calling the `EnzymeMLWriter` function (*<GitHub>/pyenzyme/Resources/Examples/ThinLayers/BioCatNet*).

3.2 Creating EnzymeML documents from STRENDA DB entries

The export from STRENDA DB follows the same concept as the conversion of a spreadsheet. The elements of the XML data model of STRENDA DB are mapped to EnzymeML objects using the a PyEnzyme object layer. Then, an EnzymeML document is created via the `EnzymeMLWriter` function.

The STRENDA DB entry 3IZNOK contains information on kinetic studies of the tryptophan biosynthesis using the TrpB2o enzyme from *Arabidopsis thaliana* (*<GitHub>/pyenzyme/Resources/Examples/ThinLayers/STRENDA/Generated/3IZNOK_TEST /3IZNOK_TEST.omex*) and was converted to an EnzymeML document (*<GitHub>/pyenzyme/Resources/Examples/ThinLayers/STRENDA/3IZNOK_TEST.xml*)

3.3 Upload of EnzymeML data to SABIO-RK

Because EnzymeML is a SBML based dialect, open-source API's like libSBML[7] and JSBML[8] were used to read, write, and edit SBML documents. In SABIO-RK, the upload and storage of data from SBML files has been already implemented as part of the data input interface to allow the upload of data from SBML models in SABIO-RK. The input interface is used to store the information provided in publications and SBML files in a structured form and to allow database curators to check and finally insert the database entries in the public SABIO-RK database. Because the existing data input interface of SABIO-RK had been written in the programming languages Groovy and Java, JSBML was used as API.

For the extraction of the data from EnzymeML, the existing parser code for uploading SBML files was extended by extracting the EnzymeML-specific annotations required for a complete SABIO-RK database entry: pH, temperature, organism name, enzyme EC-number, UniProtID, and literature reference. Before the data are transferred to the public SABIO-RK, they are manually checked, completed, and verified for possible errors and inconsistencies. Finally, the data can be retrieved from the public SABIO-RK search interface and are linked to the original reference given in the EnzymeML file (**Figure S4**).

3.4 Editing of EnzymeML: simulation of time course data from kinetic parameters

The STRENDA DB entry 3IZNOKwas used to simulate the time course of substrate at different initial concentrations in the range of 0 to 0.5 mM as noted in the STRENDA DB entry. Different initial concentrations were indicated by the `enzymeml:InitConcs` annotation for each `SpeciesReference` in the EnzymeML document. At the API level these were given as a list of float values to the `addEduct` function inherited by the `EnzymeReaction` object. The data was then simulated by applying the Michaelis-Menten model equation and the kinetic parameters (k_{cat} = 0.015 s^{-1}, K_M = 0.01 mM) over a time interval of 200 seconds. The time course data was added to the reaction object by calling the `EnzymeReaction` object method `addReplicate`, which automatically deploys the data to the respective reactant tuple, found in the list of substrates. In addition, both products L-tryptophan and HPO_4^{2-} were added via the `addProduct` function inherited by the `EnzymeReaction` function. Finally, the EnzymeML document was exported via the writer function (*<GitHub>/pyenzyme/Resources/Examples/ThinLayers/COPASI/3IZNOK_TEST/3IZNOK_TEST.omex*).

3.5 Kinetic modelling of EnzymeML data by COPASI

COPASI is a modelling platform to derive kinetic parameters from time course data.[9] In the course of this application, the generated time course data from STRENDA DB entry 3IZNOKwas imported to COPASI. First, the time course data of each replicate was exported via the `exportReplicates`-function, inherited by every reaction object, to a Pandas Dataframe. Next, the dataframe was saved to a tab-separated file and columns defined via the COPASI API (*<GitHub>/pyenzyme/Resources/Examples/ThinLayers/COPASI/3IZNOK_TEST/COPASI*). In this way, COPASI is able to map every dataset to their respective reactants. The EnzymeML document was then written to a string and parsed together with the TSV file by the COPASI API for modelling. As a result, both estimated Michaelis-Menten kinetic parameters v_{max} = 0.149 µM and K_M = 0.0099 mM as well as the equation were instantiated by the `KineticModel` class and written to the EnzymeML document via the reactions `setModel` method. Finally, the EnzymeML document was written to a file (*<GitHub>/pyenzyme/Resources/Examples/ThinLayers/COPASI/3IZNOK_TEST/COPASI/3IZNOK_TEST*). It should be noted, that alongside the programmatic parameter estimation, a .cps

file was created to be used within the COPASI GUI. Hence, a broad spectrum of models can be used to describe the kinetics.

References

1. Bergmann, F. T. *et al.* COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics* **15**, 369 (2014).
2. Favre, H. A. & Powell, W. H. *Nomenclature of Organic Chemistry*. *Nomenclature of Organic Chemistry* (Royal Society of Chemistry, 2013). doi:10.1039/9781849733069.
3. Weininger, D. SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci.* **28**, 31–36 (1988).
4. Heller, S. R., McNaught, A., Pletnev, I., Stein, S. & Tchekhovskoi, D. InChI, the IUPAC International Chemical Identifier. *J. Cheminform.* **7**, 23 (2015).
5. Novère, N. Le *et al.* Minimum information requested in the annotation of biochemical models (MIRIAM). *Nat. Biotechnol.* **23**, 1509–1515 (2005).
6. Buchholz, P. C. F., Ohs, R., Spiess, A. C. & Pleiss, J. Progress curve analysis within BioCatNet: Comparing kinetic models for enzyme-catalyzed self-ligation. *Biotechnol. J.* **14**, 1–8 (2019).
7. Bornstein, B. J., Keating, S. M., Jouraku, A. & Hucka, M. LibSBML: An API library for SBML. *Bioinformatics* **24**, 880–881 (2008).
8. Dräger, A. *et al.* JSBML: A flexible java library for working with SBML. *Bioinformatics* **27**, 2167–2168 (2011).
9. Hoops, S. *et al.* COPASI--a COmplex PAthway SImulator. *Bioinformatics* **22**, 3067–3074 (2006).

Tables

Table S1: List of attributes derived from the STRENDA recommendations

STRENDA Guidelines	**EnzymeML**
List Level 1A	
Identity of the enzyme	
Name of reaction catalyst	SBML Species:Name
EC number	EnzymeML Protein:ECNumber
Sequence accession number	EnzymeML Protein:seqAcc
Organism/species & strain	EnzymeML Protein:organism
Additional information on the enzyme	
Isoenzyme (variant)	not included
Tissue	not included
Organelle	not included
Localization	not included
Post-translational modification	not included
Preparation	
Description	not included
Artificial modification	not included
enzyme or protein purity	not included
Metalloenzyme	not included
Storage Conditions	
Storage temperature	not included
Atmosphere if not air	not included
pH	not included
At which temperature was the pH measured?	not included
Buffer & concentrations (including counter-ion)	not included
Metal salt(s) & concentrations	not included
Other components	not included
Enzyme/protein concentration	not included
Assay Conditions	
Substrate purity	not included
Measured reaction	SBML Reaction:name

Assay temperature	EnzymeML Conditions:temperature
Assay pressure	not included
Atmosphere if not air	not included
Assay pH	EnzymeML Conditions:pH
Buffer & concentrations	SBML Species:name/initialConcentration
Metal salt(s) & concentrations	SBML Species:name/initialConcentration
Other assay components	SBML Species:name/initialConcentration
Coupled assay components	not included
Substrate & concentration ranges	SBML Species:name/initialConcentration EnzymeML InitiConcs:initConc
Enzyme/ protein concentration	SBML Species:name/initialConcentration
Varied components	not included
Total assay mixture ionic strength	not included
Activity	
Initial rates of the reaction measured	SBML KineticLaw:localParameter
Enzyme activity	SBML KineticLaw:localParameter
Methodology	
Assay method	not included
Type of assay	not included
Reaction stopping	not included
Direction of the assay	not included
Reactant determined	not included
Additional material desirable	
Free metal cation	SBML:Species Modifer
Reaction equilibrium constant	SBML KineticLaw:localParameter
List Level 1B	
Required data for all enzyme functional data	
Number of independent experiments	EnzymeML:listOfMeasurements
Precision of measurement	not included
Referring to subunit or oligomeric form	not included
Data necessary for reporting kinetic parameters	
k_{cat}	SBML KineticLaw:localParameter
V_{max}	SBML KineticLaw:localParameter
k_{cat}/K_m	SBML KineticLaw:localParameter
K_m	SBML KineticLaw:localParameter

S0.5	SBML KineticLaw:localParameter
Coefficients of cooperativity	SBML KineticLaw:localParameter
How was the given parameter obtained	SBML KineticLaw:localParameter
Model used to determine the parameters	SBML KineticLaw:localParameter
Substrate inhibition (Ki value)	SBML KineticLaw:localParameter
Data required for reporting inhibition and activation data	
Time-dependence and reversibility	SBML KineticLaw:localParameter
Inhibition types (reversible, irreversible)	SBML KineticLaw:localParameter
Additional data in EnzymeML beyond STRENDA Guidelines	
Product(s)	SBML Species:Name
Time course data of substrate and product	CSV
CSV column definition	EnzymeML:format
Replicate definition	EnzymeML:replica
Amino acid sequence	EnzymeML Protein:sequence
General kinetic model	SBML KineticLaw:localParameter
InChI identifier for substrates and products	EnzymeML:inchi
SMILES identifier for substrates and products	EnzymeML:smiles
Literature reference: PubMed ID	EnzymeML:pmid
Literature reference: DOI	EnzymeML:doi
Literature reference: URL	EnzymeML:url

Table S2: SBO-terms in EnzymeML

SBO-Term	**Role in the reaction**	Notes
SBO:0000015	Substrate	
SBO:0000014	Enzyme	
SBO:0000011	Product	
SBO:0000020	Inhibitor	
SBO:0000021	Activator	
SBO:0000019	Modifier	
SBO:0000594	Neutral participant	Additives (like buffer)
SBO:0000336	Interactor	Additives
SBO:0000299	Metabolite	

Figures

```
<reaction metaid="METAID_R0" id="r0" name="steady-state kinetics with TrpB2
  <annotation>
    <enzymeml:reaction xmlns:enzymeml="http://sbml.org/enzymeml/version1">
      <enzymeml:conditions>
        <enzymeml:temperature value="303.15" unit="u3"/>
        <enzymeml:ph value="7.5"/>
      </enzymeml:conditions>
```

Figure S1: Example EnzymeML conditions annotation

```
<enzymeml:data xmlns:enzymeml="http://sbml.org/enzymeml/version1">
  <enzymeml:listOfFormats>
    <enzymeml:format id="format0">
      <enzymeml:column type="time" unit="s" index="0"/>
      <enzymeml:column replica="repl_0" species="s1" type="conc" unit="u2" index="1" initConcID="c1"/>
      <enzymeml:column replica="repl_1" species="s1" type="conc" unit="u2" index="2" initConcID="c2"/>
      <enzymeml:column replica="repl_2" species="s1" type="conc" unit="u2" index="3" initConcID="c3"/>
      <enzymeml:column replica="repl_3" species="s1" type="conc" unit="u2" index="4" initConcID="c4"/>
    </enzymeml:format>
  </enzymeml:listOfFormats>
  <enzymeml:listOfFiles>
    <enzymeml:file file="./data/data.csv" format="format0" id="file0"/>
  </enzymeml:listOfFiles>
  <enzymeml:listOfMeasurements>
    <enzymeml:measurement file="file0" id="M0" name="Measurement0"/>
  </enzymeml:listOfMeasurements>
</enzymeml:data>
```

Figure S2: Example EnzymeML annotation as described by the *ListOfReactions* tag

Data structure of EnzymeML to handle the CSV files. In the list of formats all different CSV formats are described with the order of the listed columns. In the list of files the files are connected with a format, in which the file is saved. The list of measurements lists all measurement replica in the file, which are ordered in the file vertically.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:bqbiol="http://biomodels.net/biology-qualifiers/"
    xmlns:bqmodel="http://biomodels.net/model-qualifiers/" xmlns:dcterms="http://purl.org/dc/terms/"
    xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#" xmlns:vCard4="http://www.w3.org/2006/vcard/ns#">
  <rdf:Description rdf:about="#META_UNIT_1">
    <bqbiol:is>
      <rdf:Bag>
        <rdf:li rdf:resource="https://identifiers.org/UO:0000062" />
      </rdf:Bag>
    </bqbiol:is>
  </rdf:Description>
</rdf:RDF>
```

Figure S3: Example MIRIAM RDF format of an annotated unit definition.

RDF is in subject, predicate and object order, while the about reference the subject is, the ‘bq:biol’ is the predicate and the bag with the identifiers is the object.

Pathway	pathway	0							
Reaction	indole + O-phospho-L-serine = L-Tryptophan + HPO4(2-)	0	**Transport:**	Reverse reaction					
Compounds									
Stoech.	Name	Abbr./Syn. Name	Role	Cell. Loc.	Loc. ID	Complex Protein		Comment	Comp. ID
						Prot. Identifier	Prot. Name		
1	Enzyme		Modifier-Catalyst		0				0
1	indole		Substrate		0				0
1	O-phospho-L-serine		Substrate		0				0
1	L-Tryptophan		Product		0				85
1	HPO4(2-)		Product		0				0

A

Kinetic law										
Type	steady-state kinetics with TrpB2o from Arabidopsis thaliana PConc: 10.00 uM						0			
Formula	kcat_s1*p0*s1/(km_s1+s1)						reversible			
Variables										
Name	Term		Do not replace variable in formula				Comment			
Parameter										
Name	Role	Type	Species	Value start	Value end	Deviat.	Unit	Unit ID	Unit def.	Comment
p0	Variable	concentration	Enzyme	10.0			µM	3	%	
s1	Variable	concentration	O-phospho-L-serine	0	5		mM	29	%	
s0	Variable	concentration	indole	100			µM	3	%	
kcat_s1	Constant	kcat		0.015			1/s	24	%	
km_s1	Constant	Km	O-phospho-L-serine	0.01			mM	29	%	
	unknown	unknown						null	%	

B

Figure S4: Upload of the EnzymeML document 3IZNOK_TEST via the SABIO-RK data input interface, with data describing the compounds and the reaction (A) and the kinetic parameters (B). This interface is used to validate inserted data and to align it to SABIO-RK data standards and controlled vocabulary. Subsequently, the data is transferred to the public database for search and data retrieval.