Food Safety Knowledge Exchange (FSKX) Format

Software Developer Guide

Version 3.2

Matthias Filter (Chair)	German Federal Institute for Risk Assessment
Miguel de Alba Aparicio	German Federal Institute for Risk Assessment
Esther M. Sundermann	German Federal Institute for Risk Assessment
Thomas Schüler	German Federal Institute for Risk Assessment

Alumni contributors:

Marcel Fuhrmann	German Federal Institute for Risk Assessment
Sascha Bulik	German Federal Institute for Risk Assessment
Guido Correia Carreia	German Federal Institute for Risk Assessment
Alexander Falenski	German Federal Institute for Risk Assessment
Carolina Plaza-Rodriguez	German Federal Institute for Risk Assessment

Contact:

Matthias Filter (matthias.filter@bfr.bund.de)

Inhalt

1. Introduction

Food safety risk assessments, control of food production processes as well as the development of new food products are nowadays supported by the application of mathematical modelling and data analysis techniques. Major challenges in that context are the efficient exchange and an easy reusability of knowledge (including analytical data, mathematical models, and simulation results) (Wilkinson et al., 2016). In other words, knowledge should become FAIR (findability, accessibility, interoperability, and reusability) (Wilkinson et al., 2016). To ensure an efficient exchange of mathematical models and simulation results, a standardized file format is required. Filter et al. (2016) proposed for that a standardized file format called "Predictive Modelling in Food Markup Language (PMF-ML)" that is compliant to SBML. This format describes in detail how experimental data and mathematical models from the domain of food safety can be saved and encoded in a software-independent manner. A key component of PMF-ML is the option to provide metadata that is essential for an accurate and complete description of the model as well as to provide parameters used for model-based simulations.

Here, we present "Food Safety Knowledge Exchange (FSKX) Format", an advancement of PMF-ML. FSKX format allows to describe models implemented in different script-based programming languages like R, or Python. Also, FSKX format allows to define model class-specific annotation schema, i.e. different model classes from the field of food modelling can be annotated in a harmonized way. The format also describes how to encode combined models and how other model-related information (e.g. simulation results, software packages, and visualization scripts) can be included.

In summary, FSKX format aims at harmonizing the exchange of food safety knowledge (e.g. predictive models) including the corresponding metadata. The standard will allow creating information objects that can be made available in a FAIR way and is ready-to-use.

It is noteworthy, that there are already software tools that allow importing and exporting of models in the FSKX format and thus overcome an error-prone manual file generation process. Examples for such software tools are the KNIME-based FSK-Lab (de Alba Aparicio et al., 2018) and the R-extension FSK2R (https://cran.r-project.org/web/packages/FSK2R/index.html).

This guidance document is primarily designed for software developers and project managers that want to enhance their software tools with import and export functions for models, simulations, or data in the domain of food safety and risk assessment. It might also be of interest for those who develop new tools.

2. Related Standards

FSKX format is based on (1) SBML, (2) OMEX, and (3) SED-ML. This section provides an overview on these formats and also introduces "Food Safety Knowledge Simulation Experiment Description Markup Language (FSK-SED-ML)".

2.1. Systems Biology Markup Language (SBML)

The Systems Biology Markup Language (SBML) is a representation format for (biological) models that is based on XML, the eXtensible Markup Language (http://www.sbml.org/). SBML can be used by different software tools and thus facilitates to exchange the model and also ensure that the model can be used even if the software in which the model was created in is no longer available (see Hucka et al. (2015) for details). The FSKX format specified in this document uses the SBML Level 3 Version 1.

2.2. Open Modelling EXchange Format (OMEX)

The Open Modelling EXchange format (OMEX) aims to support the exchange of information necessary for modelling and simulating experiments in biology. OMEX defines an OMEX file (also called COMBINE archive) as a zip file. The zip file includes at least a file with a listing of the container content, the so called *Manifest*. In addition, it is possible for OMEX files to include any kind of additional data, e.g. files that contain metadata or a mathematical model (see Bergmann et al. (2014) for details). The FSKX format specified in this document uses OMEX Version 1.

2.1. Food Safety Knowledge Simulation Experiment Description Markup Language (FSK-SED-ML)

Food Safety Knowledge Simulation Experiment Description Markup Language (FSK-SED-ML) extends the Simulation Experiment Description Markup Language (SED-ML) to describe the simulation settings for FSKX models. SED-ML encodes simulation descriptions on computational models of biological systems (see Waltemath, Bergmann, Adams, and Le Novere (2011) for details). FSK-SED-ML is explained in detail in the Supplementary Information.

3. Document Conventions

In this section, the terminology as well as the typographical conventions for this document are presented.

3.1. Terminology

To prevent misunderstanding, relevant terms are defined in the following:

Generic Metadata Schema

The complete set of metadata concepts that allows annotating food safety models or data (definition is adapted from Haberbeck et al. (2018)).

Metadata

Data that defines and describes other data (ISO (International Organisation for Standardisation), 2004) (definition is taken from Haberbeck et al. (2018)).

Model Class

Classification of models according to their purpose, e.g. QMRA model, dose-response model, process model, consumption model, exposure model, health metric model, predictive microbial model and other empirical model. The current classification is rooted mainly on the definition of risk assessment provided by Codex Alimentarius (Codex Alimentarius Commission, 1999).

Simulation

A computer simulation is the reproduction of the behaviour of a system using a computer to represent the outcomes of a model that represent the said system (based on Wikimedia Foundation Inc. (2019)).

Note: There is a difference between a (mathematical) model and a (model-based) simulation. A (mathematical) model is composed of the algorithms and equations used to mimic the modelled system. By contrast, a (model-based) simulation is the process of running a (mathematical) model to make a prediction on the system's behaviour (based on Wikimedia Foundation Inc. (2019)).

3.1. Implementation Convention

In this section, the implementation conventions for the format are defined.

3.1.1. Type SId

In SBML, there is the so called SId type; FSKX format also uses this type for certain metadata related to SBML and SED-ML. SId type restricts the characters permitted and the sequences in which those characters are allowed to appear. The definition is shown in Figure 1 (for more details see e.g. Hucka et al. (2015)).

```
letter   ::=   ‘a’..’z’,’A’..’Z’
digit    ::=   ‘0’..’9’
idChar   ::=   letter | digit | ‘_’
SId      ::=   ( letter | ‘_’ ) idChar*
```

Figure 1 Definition of the type SId (the figure is taken from Hucka et al. (2015)).

3.1.2. Script Language Specificity

In FSKX format, models and visualisation scripts can be script language-specific, i.e. these scripts can be provided in programming languages like R or Python. For simplicity, examples presented in the following are always based on the R scripting language (R Core Team, 2019).

3.1.3. Referencing Script Code

In FSKX format, there are two ways to refer to information that is provided as script code, e.g. a model implemented in R. First, the script code can be provided directly. This option only becomes relevant for the joining of multiple models (see Section 5.1.1 for details). Alternatively, the path to a script file can be provided (see Supplementary Information for details).

3.2. Typographical Conventions

This document uses the typographical conventions defined in the OMEX specification document (Bergmann et al., 2014). Names of objects, classes, and data types are highlighted as follows:

Class: Names of ordinary (concrete) classes begin with a capital letter and are printed in an upright, bold, and sans-serif typeface. In electronic document formats, the class names defined within this document are also hyperlinked to their definitions; clicking on these items will, given appropriate software, switch the view to the section in this document containing the definition of the class. In this document, class names are hyperlinked to the definition in this document only if they are changed from their definitions in the SED-ML Level 1 Version 1 specification (Waltemath et al., 2011).

AbstractClass: Abstract classes serve as parents of other classes. Their names begin with a capital letter and they are printed in a slanted, bold, and sans-serif typeface. In electronic document formats, the class names defined within this document are also hyperlinked to their definitions; clicking on these items will, given appropriate software, switch the view to the corresponding section in this document. Note that, for classes that are defined in SED-ML Level 1 Version 1 specification (Waltemath et al., 2011), the class names are not hyperlinked because they are not defined within this document.

`OtherAttributes`: Attributes of classes, data type names, literal XML, and tokens are printed in an upright typewriter typeface. Note that this convention is not meant for class names.

4. Specification: Single FSKX-Model

In this chapter, the core specifications of FSKX format are presented focusing on single models. The specifications for data and the combination of multiple single models, so called joined models, are presented in Chapters 5 and 6, respectively.

The FSKX format describes how relevant information should be provided, i.e. what files are needed or possible to describe content relevant for food safety modelling. We distinguish the following file type purposes:

(e) file(s) containing input information, i.e. this needs to be provided by the FSKX model creator or contain information that is relevant for successful **e**xecution of the model, and

(u) file(s) containing information that supports the **u**ser, but is not relevant for the model execution.

Table 1 lists all files that are mandatory or recommended to be included to comprehensively describe a model. In order to exchange all files as one information object, a container format is proposed, called FSKX-container. An FSKX-container is a zip file containing at least one file, the mandatory *manifest.xml* file. The *manifest.xml* file lists all files inside of the FSKX-container, i.e. it can include any number of files (see Section 4.1.1 for details). In Table 1, examples for file names and file extensions are listed. Note that other file extension are possible, e.g. a model script in R could save results in an Excel file and thus, *sim_1_res.xlsx* is another possibility to store simulation results. We recommend the file extension ".fskx" for the FSKX-container.

Table 1 A listing of (mandatory and recommended) files that can be included into an FSKX-container. For each file the desciption, example file names, information about whether or not a file is mandatory, and where to find futher details are given.

	Description	Example file name and file extension	Is the file mandatory?	Purpose of the file*	Further details
Container structure related files	List of all files in the FSKX-container	manifest.xml	yes	e	See Section 4.1.1
	Additional information about the files in the FSKX-container	metadata.rdf	yes	e	See Section 4.1.2
Model related files	Model script	model.r model.py model.php	yes	e	See Section 4.2.1
	Metadata about the used model	metadata.json	yes	e	See Section 4.2.2
	Model annotation template	modelann.xlsx	no	u	See Section 4.2.2
	Supplements to the model, e.g. the paper where the model is published	description.pdf	no	u	
	Model related data	experiment.xlsx experiment.csv	no	e	See Section 4.2.3
	Required software and packages	pack_0.10.zip pack_0.10.tgz pack_0.10.tgz-gz packages.json	no	e	See Section 4.2.4
Simulation related files	Simulation settings	sim_1.sedml	yes	e	See Section 4.3.1
	SBML file that contains the simulation settings	model.sbml	recommende d	e	See Section 4.3.1
	Simulation results	sim_1_res.RData sim_1_res.csv sim_1_res.json	no	u	See Section 4.3.2
	Visualization script	visualization_1.r visualization_1.py	no	e	See Section 4.3.3

	Plots of the simulated results	plot.png, plot.bmp plot.tiff	no	u	See Section 4.3.2
Other	Read me	README.txt	yes	u/e	

*file contains input or is relevant for the execution of the model (e), or file supports users (u)

Figure 2 shows an example file structure of an FSKX compliant model, i.e. it indicates the position of files within folders. This structure is not mandatory but highly recommended.

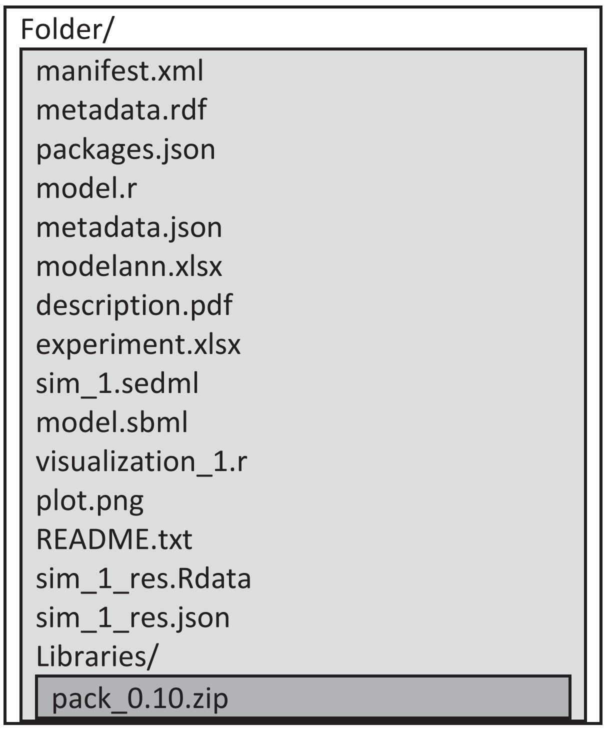


Figure 2 Recommended container structure of an FSKX compliant model.

4.1. Container Structure Related Files

In this section, the files that describe the structure and content of the FSKX-container are elucidated.

4.1.1. Manifest

The manifest file lists all files inside the FSKX-container (*manifest.xml* in Figure 2). That means that a valid manifest file needs to have at least one entry, declaring the container itself. For an example of a manifest file, see Figure 3 (see Section 7.1 for details about the example).

```
<?xml version="1.0" encoding="UTF-8"?>
<omexManifest xmlns="http://identifiers.org/combine.specifications/omex-manifest">
  <content location="." format="http://identifiers.org/combine.specifications/omex" />
  <content location="./manifest.xml"
format="http://identifiers.org/combine.specifications/omex-manifest" />
  <content location="./model.sbml" format="http://purl.org/NET/mediatypes/application/sbml+xml"
/>
  <content location="./plot.png" format="http://purl.org/NET/mediatypes/image/png" />
  <content location="./visualization.r" format="http://purl.org/NET/mediatypes/application/r"
/>
  <content location="./sim.sedml" format="http://identifiers.org/combine.specifications/sed-ml"
/>
  <content location="./defaultSimulation.r"
format="http://purl.org/NET/mediatypes/application/r" />
  <content location="./workspace.RData" format="http://purl.org/NET/mediatypes/text/x-RData" />
  <content location="./metaData.json" format="https://www.iana.org/assignments/media-
types/application/json" />
  <content location="./model.r" format="http://purl.org/NET/mediatypes/application/r" />
  <content location="./packages.json" format="https://www.iana.org/assignments/media-
types/application/json" />
  <content location="./README.txt" format="http://purl.org/NET/mediatypes/text-xplain" />
  <content location="./simulations/defaultSimulation.r"
format="http://purl.org/NET/mediatypes/application/r" />
  <content location=".\metadata.rdf"
format="http://identifiers.org/combine.specifications/omex-metadata" />
</omexManifest>
```

Figure 3 Example for a manifest file in XML.

The manifest file is an XML file and is located at the root of the container. This file contains an instantiation of the *OmexManifest class* (already defined in Bergmann et al. (2014)). For each file that is in the container there is a URI given to specify the file type. This specification is done by URIs through the Internet media types, which is previously known as MIME type (Freed & Borenstein, 1996). Table 2 shows a list of relevant file types and associated Internet media types.

Table 2 Relevant file types and corresponging Internet media types that can be used in the manifest file. Note that the Internet media types are identifiers and no links.

File type	Internet media type
Zip	http://purl.org/NET/mediatypes/application/zip
Tgz	http://purl.org/NET/mediatypes/application/x-tgz
Tag-gz	http://purl.org/NET/mediatypes/application/x-tar.gz
R	http://purl.org/NET/mediatypes/application/r
Python	http://purl.org/NET/mediatypes/application/python
PMF	http://purl.org/NET/mediatypes/application/x-pmf
SBML	http://purl.org/NET/mediatypes/application/sbml+xml

JSON	https://www.iana.org/assignments/media-types/application/json
Matlab	http://purl.org/NET/mediatypes/text/x-matlab
PHP	http://purl.org/NET/mediatypes/text/x-php
Plain text	http://purl.org/NET/mediatypes/text-xplain
R workspace	http://purl.org/NET/mediatypes/text/x-RData
CSV	https://www.iana.org/assignments/media-types/text/csv
Sedml	http://identifiers.org/combine.specifications/sed-ml
XLSX	https://www.iana.org/assignments/media-types/application/vnd.ms-excel
BMP	https://www.iana.org/assignments/media-types/image/bmp
JPEG	https://www.iana.org/assignments/media-types/image/jpeg
TIFF	https://www.iana.org/assignments/media-types/image/tiff
PNG	http://purl.org/NET/mediatypes/image/png
HDF5	http://purl.org/NET/mediatypes/application/x-hdf5

4.1.2. Metadata File

The metadata file specifies files that are used or generated by the model execution engine. Specifically, it is needed to distinguish the files holding the mathematical model, the visualization script, and in some cases files storing simulation results. The model inclusion has to be done in one out of two ways (see Section 3.1.3 for details). Note that simulation scenarios (with all corresponding input parameter values) are stored in a software-independent representation as FSK-SED-ML file (see Section 4.3.1 for details).

The metadata file uses the Resource Description Format (RDF; *metadata.rdf* in Figure 2; see Cyganiak et al. (2014) for details about RDF). The version of the RDF file is described by the `conformsTo`-property from the Dublin Core Metadata Initiative (see Figure 4 for an example and Section 7.1 for details about the example; see https://www.dublincore.org/ for further details about the Dublin Core Metadata Initiative). The RDF-annotation involves RDF description elements that specify the type and location of the resource with Dublin Core `type` and `source` elements, respectively. Table 3 lists the types of the RDF file, their description, and the information whether or not the type is mandatory.

Table 3 Types, the description, and the information whether or not the type need to be specified in the RDF file.

Types	Description	Is the type mandatory?
annotationSheet	Filename of an Excel spreadsheet that contains model metadata. Some software tools allow to provide model metadata via an Excel spreadsheet (i.e. this Excel spreadsheet is the initial provision of metadata that might be adjusted using the software tool and then, is transformed into the mandatory *JSONMetaData* JSON file). These tools provide the original Excel files also	no

	inside the FSKX folder (as a backup). It should be noted that no congruency between the annotation in the Excel spreadsheet and the mandatory machine-readable *JSONMetaData* JSON file can be assumed.	
JSONOutput	Filename of the JSON file that contains the results of the model simulation, i.e. it contains the value for each output parameter (see Section 4.3.2 for details).	no
JSONMetaData	Filename of the JSON file that contains the metadata (see Section 4.2.2 for details)	yes
mainScript	Filename of the main model script (see Section 4.2.1 for details). Note, in the case only one model script is provided, the types *mainScript* and *modelScript* are accepted. In the case multiple scripts are available, the file that is classified to be the *mainScript* is executed first.	no
modelScript	Filename of any other model script that might be sourced from the main script (see Section 4.2.1 for details). Note, in the case only one model script is provided, the types *mainScript* and *modelScript* are accepted types.	no
readme	Filename of the readme file.	yes
visualizationScript	Filename of the visualization script (see Section 4.3.3 for details).	no
workspace	Filename of the workspace with the simulation results (see Section 4.3.2 for details).	no

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF                                xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms="http://purl.org/dc/terms/" xmlns:vCard="http://www.w3.org/2006/vcard/ns#">
  <rdf:Description rdf:about=".">
    <dcterms:conformsTo>2.0</dcterms:conformsTo>
  </rdf:Description>
  <rdf:Description rdf:about="/visualization.r">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">visualizationScript</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/workspace.RData">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">workspace</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/model.r">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">modelScript</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/README.txt">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">readme</dc:type>
  </rdf:Description>
</rdf:RDF>
```
Figure 4 Example for a metadata file in the RDF-format.

4.2. Model Related Files

This section specifies the files that are related to the presented model.

4.2.1. Model Script

The model script is a file that contains the model code. The code can be programming language-specific, e.g. in R or Python (*model.r* in Figure 2). It is possible to store multiple model scripts within the FSKX-container.

4.2.2. Model Metadata

The model metadata must be provided in a *metadata.json* file (see Figure 2). The JSON file is structured according to the metadata concepts and controlled vocabularies that are defined by the so called Generic Metadata Schema. This metadata schema and the linked controlled vocabularies are provided as a community driven, online resource and thus is subject to regular improvements (see Metadata Master Table and controlled vocabularies on https://foodrisklabs.bfr.bund.de/rakip-harmonization-resources/). For details about the JSON file, see "Metadata schema" on https://foodrisklabs.bfr.bund.de/fskx-food-safety-knowledge-exchange-format/ or https://github.com/SiLeBAT/fskml_schemas/blob/main/model.yaml. The Generic Metadata Schema also provides specific metadata sets for certain model classes. Table 4 lists a description of the currently defined model classes and the name of the corresponding sheet in the Metadata Master Table. Note, for models that can't be annotated with any of the specific metadata sets, the metadata defined in the Generic Metadata Schema sheet are to be used. The minimum information that need to be provided in the JSON file have to fulfil the MIRARAM guidelines (Filter et al., 2021). In the Generic Metadata Schema, the cardinality of each metadata field shows whether or not the information is mandatory.

Table 4 List of currently supported model classes, a desciption, and the name of the corresponding sheet in the metadata file (effective December 2020).

Model class	Description	Sheet name
Generic model	A model class that is generic and that serves as the common denominator for all explicit model classes. This means that for each metadata field in any of the explicit model classes there must be a 1-on-1 mapping to a metadata field in this generic model sheet.	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata_-Master_Table_V1.04.xlsx (sheet name: Generic Metadata Schema; on Sheet 1)

Consumption model	A consumption model describes the amount of food consumed during a particular eating occasion (i.e., a serving) and/or the frequency of the consumption of these servings, or an average amount of food consumed per day. This amount may vary in time, between individuals, between the different population groups of interest and the considered exposure type (definition is taken from Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata_-Master_Table_V1.04.xlsx (sheet name: Consumption Model; on Sheet 9)
Data	Symbolic representation of observable properties of the world (definition is taken from Haberbeck et al. (2018)). See Chapter 6 for details.	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata_-Master_Table_V1.04.xlsx (sheet name: (Data)) on Sheet 2
Dose-response model	Model describing the "relationship between the magnitude of exposure (dose) to a hazard and the severity and/or frequency of associated adverse effects (response)" (Codex Alimentarius Commission, 1999; FAO/WHO, 2016) (definition is taken from Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata_-Master_Table_V1.04.xlsx (sheet name: Dose-response Model; on Sheet 5)
Exposure model	A combination of the process model and the consumption model that results in the exposure assessment (definition is taken from Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata_-Master_Table_V1.04.xlsx (sheet name: Exposure Model; on Sheet 6)
Health metrics model	Model for calculating a measure for assessing the health impact of a specific hazard in a population group: e.g. Disability-adjusted life years (DALYs) or cost per illness (definition is taken from Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata_-Master_Table_V1.04.xlsx (sheet name: Health metrics Model; on Sheet 10)
Other empirical model	A generic model that describes a set of data in a convenient mathematical relationship without considering any underlying phenomena (definition is based on Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata_-Master_Table_V1.04.xlsx (sheet name: Other Empirical Model; on Sheet 4)
Predictive model	Models describing the microbial responses towards environmental conditions, such as storage and	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metada

	processing conditions and product characteristics. Traditionally, models in predictive microbiology are classified as primary and secondary (Whiting, 1993) (definition is taken from Haberbeck et al. (2018)). For further details, see Haberbeck et al. (2018).	ta -Master Table V1.04.xlsx (sheet name: Predictive Model; on Sheet 3)
Process model	Model that describes how the concentrations of the hazard change along the different steps (modules) of the food production chain (potentially from farm to fork) (definition is taken from Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata -Master Table V1.04.xlsx (sheet name: Process Model; on Sheet 9)
Quantitative risk assessment (QRA) model	The quantitative modelling of a scientifically based process consisting of the following steps: (i) hazard identification, (ii) hazard characterization, (iii) exposure assessment, and (iv) risk characterization (Codex Alimentarius Commission, 1999; FAO/WHO, 2016) (definition is based Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata -Master Table V1.04.xlsx (sheet name: QRA Model; on Sheet 12)
Risk characterization model	Combination of health metrics model, dose-response model and exposure model within the framework of the risk characterization (definition is taken from Haberbeck et al. (2018)).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata -Master Table V1.04.xlsx (sheet name: Risk characterization model; on Sheet 11)
Toxicological reference value model	Modelling a value, that when compared with exposure, is used to estimate the likelihood and severity of an adverse effect which could occur in a given population. (based on https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/RAKIP Terms-and-concepts Glossary V1.00.xlsx).	https://foodrisklabs.bfr.bund.de/wp-content/uploads/2020/11/Metadata -Master Table V1.04.xlsx (sheet name: Toxicological reference value M; on Sheet 8)

As described in Haberbeck et al. (2018), the Generic Metadata Schema is organized in a hierarchical structure with four top level elements: (1) General information, (2) Scope, (3) Data_background, and (4) mathematical model and data (so called “Model math/Data definition"). For details, see Haberbeck et al. (2018) or the online version of the Generic Metadata Schema available in the Section "Food Safety Knowledge Metadata" on https://foodrisklabs.bfr.bund.de/rakip-harmonization-resources/. As the metadata schema is a community driven resource, it is updated on a regular basis. Thus, it is recommended to provide the used Metadata schema version number within the *metadata.json* file.

Controlled Vocabularies

It is recommended to use controlled vocabularies for relevant metadata (see https://knime.bfr.berlin/vocabularies_frontend/frontend.html for the online resource, Supplementary Information for a list, or Section "Food Safety Knowledge Metadata" on https://foodrisklabs.bfr.bund.de/rakip-harmonization-resources/). The vocabularies provided there were collected from existing ontologies, standards, and tools (for more information see https://foodrisklabs.bfr.bund.de/rakip-harmonization-resources/).

To allow software tools to execute FSKX-models that are composed of modules (see Section 5), where each module was created in a different scripting language, it is essential that at least the data type for presented in Table 5 are supported. Note, the entries in this controlled data type vocabulary are case-insensitive.

Table 5 Data types that must be supported by all FSKX compliant tools.

Controlled vocabulary	Description
number	Integers and floating point numbers
string	String or character values
file	Name of a file including file extension
array	One & two-dimensional arrays
table	Annotated tables (e.g. R data.frame or pandas.Dataframe)
list	List of objects (of supported data types)

4.2.3. Model Related Data

A model might need data from separate files for model execution and to ensure transparency. Those data might be used as an input for the model, for model generation (e.g. through fitting), or for model validation. The FSKX format supports the provisioning of an unlimited number of data files (e.g. *experiment.xlsx* in Figure 2).

4.2.4. Required Software and Packages

The information about the software and the packages as well as the considered versions is crucial to execute a given model and visualization script.

The mandatory model metadata field "Language written in" specifies the required software and the version of this software (see Section 4.2.2 for details about the metadata schema and corresponding files). If multiple software tools are needed to execute the model, each software has to be listed in the "Language written in" metadata field. One example for a single model that requires multiple software tools is a Bayesian model that is implemented in R (R Core Team, 2019) and uses the software OpenBUGS to estimate parameters (Lunn, Spiegelhalter, Thomas, & Best, 2009; Neal, 2021). The information about the software should be given in a JSON file as well (*packages.json* in Figure 2). This JSON file should describe the programming language and has to specify the used packages (see Figure 5 for an example; see Section 7.1 for details about the example). The format of the JSON file is described in detail in the Section "Packages schema" on https://foodrisklabs.bfr.bund.de/fskx-food-safety-knowledge-exchange-format/. It is advisable, but not mandatory, to also include those packages that show dependencies to the listed packages.

It is recommended to add the libraries of the packages to the FSKX-container. Although the placement of the libraries in the FSKX file is free, it is advisable to save them into a separated *Libraries* folder (*pack_0.10.zip* in Figure 2). It is acknowledged, that it cannot be assumed that provided packages run on all operating systems. Instead, it is assumed that the model creator only provides the packages that correspond to the operating system that was used during creation of the FSKX file. For example, in the case of an R-model created under Windows 10, the corresponding packages would be windows binary packages.

```
Language      "R"
PackageList   []
```

Figure 5 Example for a JSON file that contains information about the programming language and used packages.

4.3. Simulation Related Files

FSKX format not only provides an opportunity to describe script-based models, it also enables to define, store, and exchange settings for model-based simulations in a **software-independent** manner. All relevant settings, e.g. for model input parameters, are stored in FSK-SED-ML format (see Section 4.3.1 for details). The results from executed simulation scenarios can be shared as well (see Section 4.3.2 for details about storage and Section 4.3.3 for details about the visualization of results). The FSK-SED-ML format is described in more detail in the Supplementary Information.

4.3.1. Simulation Settings

The FSK-SED-ML file (*sim_1.sedml* in Figure 2) contains simulation settings, i.e. the parameter set (including input parameters and variables) and relevant metadata, e.g. a name and a description of each simulation scenario. This file contains at least one parameter set for the so-called “Default” scenario. It can also contain multiple parameter sets for all scenarios defined by the model creator. In addition to the FSK-SED-ML files, an SBML file should be provided, that contains the default parameters for user convenience (see Table 1 and *model.sbml* in Figure 2). This file is recommended, because it ensures compatibility between single and joined models (see Chapter 5 for details about the joined models).

Note that each model creator should carefully select the input parameters of the model. Dependencies of one input parameter to another parameter might lead to issues during model execution as the execution order of parameter value assignments is defined by the order as the parameter are listed inside the FSK-SED-ML file!

4.3.2. Simulation Results

The results for each simulation can on the one hand be stored in form of figures and on the other hand in form of files. In general, it is advisable to save the simulation results in a widespread format; we recommend the JSON-format (*sim_1_res.json* in Figure 2). The schema how to store the value for each output parameter is described in detail the Section “Parameter schema**”** on https://foodrisklabs.bfr.bund.de/fskx-food-safety-knowledge-exchange-format/. If the model is implemented in R, simulation results can also be saved as an R-workspace (*sim_1_res.Rdata* in Figure 2). Common formats in which generated plots can be saved are JPEG, PNG, and SVG files (*plot.png* in Figure 2). Although there is no restriction where simulation results should be placed within the FSKX-container, it is recommended to save them in a separated *Simulation* folder.

4.3.3. Visualization Scripts

Visualization scripts create plots from the simulation results (*visualization_1.r* in Figure 2). Plots might also be a composition of multiple sub-figures. The script has to be in the same scripting language as the corresponding model. See Section 4.3.2 for details about the storage of simulation results.

5. Specification: Joined FSKX-Model

This section describes joined models in an FSKX compliant way. Joined models are models where at least two independent FSKX-models are combined into one FSKX file. Usually, a joined model links output parameter(s) of one model (donor model) to input(s) of a second model (receiver model).

An FSKX compliant joined model is represented in multiple files; maintaining consistency with the single model structure (see Chapter 4). Identical to the single model structure (see Table 1), we distinguish two purposes for the files: **(e)** the file contains input or the file contains information that is relevant for the **e**xecution of the model and **(u)** the file contains information that support the **u**ser. Table 6 lists mandatory and recommended files for the joined model representation in the FSKX format. For each file, the description, example filenames, and information about whether or not a file is mandatory are presented.

The information on how model parameters are linked between two models is specified in an SBML file, which is located at the root directory of the FSKX file (see Section 5.1.1).

Please note:

1. FSKX files can contain a number of nested joined models.
2. The description is structured such that two models are joined at a time. This means that nested models with more than two models will be described by multiple consecutive joining steps (for a combination of n models, n-1 joining steps are required).

Table 6 A list of (mandatory and recommended) files that can be in the FSKX-container of a joined model. For each file the description, example file names, and information about whether or not a file is mandatory is given.

	Description	**Example file name with file extension that are supported**	**Is the file mandatory?**	**Purpose of the file***	**Further details**
Joining related files	The SBML file that describes the joining of the models	joined_model.sbml	yes	e	See Section 5.1.1
	Metadata about the joined model	metadata.json	yes	e	See Section 5.1.2
	Simulation settings for the joined model	sim_1.sedml	yes	e	See Section 5.1.3
Contai ner	List of all files in the FSKX-container	manifest.xml	yes	e	See Section 5.2.1

	Additional information about the files in the FSKX-container	metadata.rdf	yes	e	See Section 5.2.2
	Required software and packages	packages.json	no	u	See Section 5.2.3
Single model related files	One folder for each model. The folder complies to the structure for a single model (except for the manifest (.xml) and the metadata container (.rdf); see Table 1 for the single model elements)	Folder/	yes	e	See Section 5.3

*file contains input or file is relevant for the execution of the model (e), or file supports user (u)

Figure 6 represents the container structure of an FSKX compliant joined model, i.e. it indicates the position of files within folders and subfolders. This structure is mandatory. Pease not that in the case of multiple nested model the at least one of the single model folder (*DonorModelFolder* or *ReceiverModelFolder*) contains the folder *JoinedModelFolder* and all the subfolder of the first joining models.

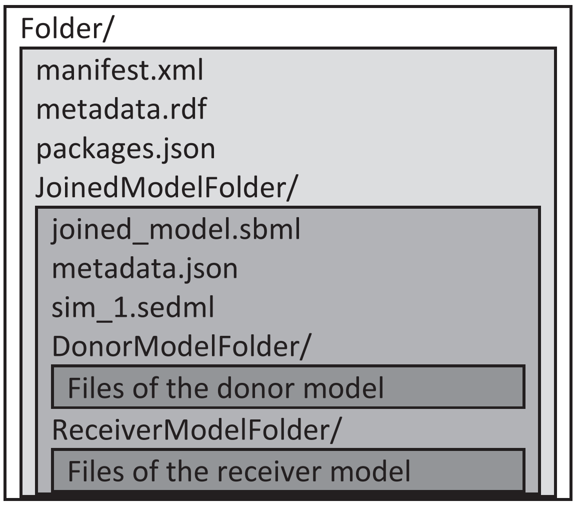


Figure 6 Example container structure of an FSKX compliant model that is joined from two original models.

5.1. Files Related to Model Joining

In this section, the files that describe how two models are combined are presented.

5.1.1. Joined Model Declaration File

The information how to join two models is provided inside a software-independent SBML file (*joined_model.sbml* in Figure 6). In this file, the receiver and the donor model are specified (for an example see *ExampleExposureModel.sbml* and *ExampleDoseResponseModel.sbml* in Figure 7; that represents a fictive example) as well as the linkage between parameters from both models (the value of *doseValue2* is set to *dose1* in Figure 7).

```
<?xml version='1.0' encoding='UTF-8' standalone='no'?>
<sbml     xmlns="http://www.sbml.org/sbml/level3/version1/core"     comp:required="true"
level="3" version="1" xmlns:comp="http://www.sbml.org/sbml/level3/version1/comp/version1"
xmlns:fsk="https://foodrisklabs.bfr.bund.de/wp-content/uploads/2017/01/FSK-
ML_guidance_document_021216.pdf">
  <comp:listOfExternalModelDefinitions
xmlns:comp="http://www.sbml.org/sbml/level3/version1/comp/version1">
    <comp:externalModelDefinition                          comp:id="ExampleExposureModel"
comp:source="ExpJoinedModel\ExampleExposureModel\ExampleExposureModel.sbml"/>
    <comp:externalModelDefinition                      comp:id="ExampleDoseResponseModel"
comp:source="ExpJoinedModel\ExampleDoseResponseModel\ExampleDoseResponseModel.sbml"/>
  </comp:listOfExternalModelDefinitions>
  <model id="ExpJoinedModel">
    <comp:listOfSubmodels
xmlns:comp="http://www.sbml.org/sbml/level3/version1/comp/version1">
      <comp:submodel comp:id="submodel1" comp:modelRef="ExampleExposureModel"/>
      <comp:submodel comp:id="submodel2" comp:modelRef="ExampleDoseResponseModel"/>
    </comp:listOfSubmodels>
    <listOfParameters>
      <parameter constant="false" id="doseValue2">
        <annotation>
  <fsk:command commandValue="[dose1]"/>
        </annotation>
              <comp:replacedBy
xmlns:comp="http://www.sbml.org/sbml/level3/version1/comp/version1"    comp:idRef="dose1"
comp:submodelRef="submodel1"/>
      </parameter>
    </listOfParameters>
  </model>
</sbml>
```

Figure 7 Example for the SBML file that describes the joining of two models.

5.1.2. Model Metadata

The *metadata.json* file (see Figure 6) contains all metadata that describe the joined model. As a joined model is often composed of models that belong to different model classes, the joined model is usually annotated with the Generic Metadata Schema (see Section 4.2.2 for further details).

5.1.3. Simulation Settings

The FSK-SED-ML files contains simulation settings for the joined model (see Section 4.3.1 for details; *sim_1.sedml* in Figure 6).

5.2. Container Structure Related Files

In this section, the files that describe the structure and content of the FSKX-container are elucidated in more detail.

5.2.1. Manifest

The manifest file lists all files inside the joined model FSKX-container (including those from the donor and receiver model) (see Section 4.1.1 for details; *manifest.xml* in Figure 6).

5.2.2. Metadata File

The metadata file identifies the model scripts, visualization scripts and files that store simulation results (see Section 4.1.2 for details; *metadata.rdf* in Figure 6). Figure 8 shows an example for the RDF file of a joined model (see Section 5.2.2 for details about the example). The model scripts of the single models are classified using the *modelScipt* type (that is described in Table 3).

```
</rdf:RDF>
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF                                xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dcterms="http://purl.org/dc/terms/" xmlns:vCard="http://www.w3.org/2006/vcard/ns#">
  <rdf:Description rdf:about=".">
    <dcterms:conformsTo>2.0</dcterms:conformsTo>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleDoseResponseModel/visualization.r">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">visualizationScript</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleDoseResponseModel/workspace.RData">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">workspace</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleExposureModel/workspace.RData">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">workspace</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleDoseResponseModel/README.txt">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">readme</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleExposureModel/README.txt">
```

```
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">readme</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleDoseResponseModel/model.r">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">modelScript</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/workspace.RData">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">workspace</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleExposureModel/visualization.r">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">visualizationScript</dc:type>
  </rdf:Description>
  <rdf:Description rdf:about="/ExpJoinedModel/ExampleExposureModel/model.r">
    <dc:type xmlns:dc="http://purl.org/dc/elements/1.1/">modelScript</dc:type>
  </rdf:Description>
</rdf:RDF>
```

Figure 8 Example for a metadata file in the RDF format that describes a joined model.

5.2.3. Required Software and Packages

Information about software and software packages required to execute any of the models (donor and receiver model) or to visualize results have to be listed in the FSKX-container.

The information on the programming language used for the model has to be provided as part of the model annotation. The mandatory metadata field “Language written in” specifies this information (see Section 4.2.2 for details about the metadata schema). If models that are implemented in various languages are joined, all the languages have to be listed in this field. Also, this information must be included in the SEDML file (see Section 4.3.1 for details).

The information about the packages have to be presented in two ways. First, the model specific information are provided for donor and receiver model separately and in the way that is described for the single model (see Section 4.2.4 for details). Second, the information for both, the donor and the receiver model, are provided in a JSON file at the root of the folder. This JSON file is an array that has to list all packages and should list the programming language (see Figure 9 and Figure 10 for examples; the example presented in Figure 9 is fictive). The schema of the JSON file is described in detail in the Section “Packages schema” on https://foodrisklabs.bfr.bund.de/fskx-food-safety-knowledge-exchange-format/.

```
[
    {
        "modelId": "ExpDR",
        "language": "R",
        "languageVersion": "3",
        "packageList": [ ]
    },
    {
        "modelId": "ExpEpo",
        "language": "R",
        "languageVersion": "3",
        "packageList": [  ]
    }
]
```
Figure 9 Example for the package.json for an fictive example of a joined model.

```
[
    {
        "modelId": "model1",
        "language": "R",
        "languageVersion": "3.4.4",
        "packageList": [
            {
                "package": "gridExtra",
                "version": "2.3"
            },
            {
                "package": "MALDIquant",
                "version": "1.19.3"
            }
        ]
    },
    {
        "modelId": "model12",
        "language": "Python",
        "languageVersion": "3.7",
        "packageList": [
            {
                "package": "pandas",
                "version": "1.1.3"
            },
            {
                "package": "numpy",
                "version": "1.19.2"
            }
        ]
    }
]
```
Figure 10 Example for the package.json for a joined model that comprises single models implemented in different programming languages.

5.3. Files Related to the Donor and Receiver Model

The files that describe the receiver and donor model equal the ones for the single model presented in Table 1 (except for the manifest (.xml) and the metadata container (.rdf) which are not included). See Table 1 for the single model elements and see Chapter 4 for details.

6. Specification: FSKX-Data Files

This section specifies how data and data sets from the domain of (microbial) food safety can be exchanged in an FSKX compliant way. Note, the FSKX-data file specification allows to share scripts that can be used or are necessary for data visualization or data transformation (those scripts are not considered mathematical models in a strict sense). The description focuses on data that result from model simulation, but it is assumed that it is applicable to other data, too.

In order to combine all files relevant to describe data, an FSKX-container is created (see Section 4.1 for details about the FSKX-container; see Table 7 for a list of files). We distinguish two purposes for the files listed in Table 7: file(s) containing input information, i.e. this needs to be provided by the FSKX model creator, or contain information that is relevant for successful **e**xecution of the model (Purpose e) and file(s) containing information that supports the **u**ser, but is not relevant for the model execution (Purpose u).

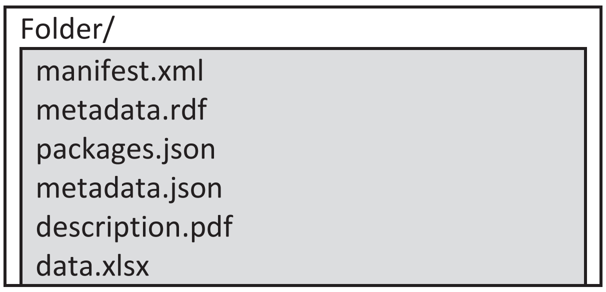
Table 7 A listing of (mandatory and recommended) files that can be included into an FSKX-container. For each file the desciption, example file names, information about whether or not a file is mandatory, and where to find futher details are given.

	Description	Example file name with file extension that are supported	Is the file mandatory?	Purpose of the file*	Further details
Container structure related files	List of all files inside the FSKX-container	manifest.xml	yes	e	See Section 6.1.1
	Additional information about the files in the FSKX-container	metadata.rdf	yes	e	See Section 6.1.2
Data related file	The file contains data (sets)	dataset.csv dataset.xlsx dataset.json	yes	e	See Section 6.2.1
	Metadata about the data	metadata.json	yes	e	See Section 6.2.2
	Supplements to the data, e.g. the paper where the data is published	description.pdf	no	u	
Visualiza tion related files	Visualization results	viz_1_res.RData viz_1_res.csv	no	u	See Section 6.3.1

	Visualization script	visualization_1.r visualization_1.py	no	e	See Section 6.3.2
	Plots of the processed data	plot.png, plot.bmp plot.svg	no	u	See Section 6.3.2
	Required software and packages	pack_0.10.zip pack_0.10.tgz pack_0.10.tgz-gz packages.json	no	u	See Section 4.2.4
Data transformation related files	Data transformation script	transform.r transform.py transform.php	no	e	See Section 6.3.2
	Supplements to the transformation steps, e.g. the paper where the model is published	description.pdf	no	u	
Transformation related files	Transformation settings	sim_1.sedml	no	e	See Section 6.4
	SBML file that contains the transformation settings	model.sbml	no	e	See Section 6.4
Other	Read me	README.txt	yes	u/e	

*file contains input or file is relevant for understanding of the data set(s) (e), or file supports user (u)

Figure 11 shows an example file structure of an FSKX compliant model, i.e. it indicates the position of files within folders and subfolders. This structure is not mandatory but highly recommended.



```
Folder/
  manifest.xml
  metadata.rdf
  packages.json
  metadata.json
  description.pdf
  data.xlsx
```

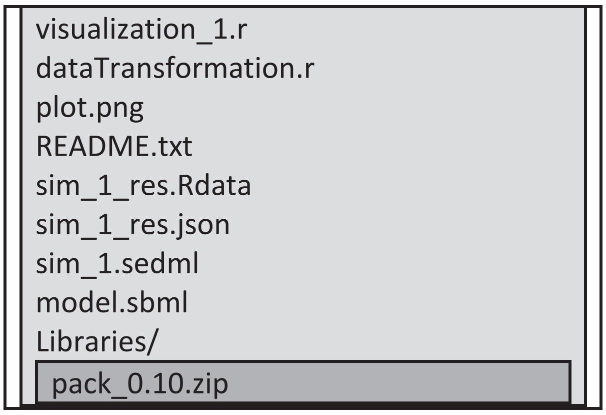


Figure 11 Example for the container structure of FSKX compliant data set.

6.1. Container Structure Related Files

In this section, the files that describe the structure and content of the FSKX-container are elucidated in more detail.

6.1.1. Manifest

The manifest file lists all files inside the FSKX-container (see Section 4.1.1 for details; *manifest.xml* in Figure 11).

6.1.2. Metadata File

The RDF-metadata file identifies the files relevant for model execution and annotation (see Section 4.1.2 for details; *metadata.rdf* in Figure 11). To incorporate the visualization script and the transformation script into the RDF file, the types *visualizationScrip* and *mainScript* are needed, respectively (for details of the types see Table 3).

6.2. Data Related Files

In this section, the files that hold the data and its annotation are presented in more detail.

6.2.1. Data Storage

Data in form of tables, multidimensional arrays, and multiple data sets can be stored. Various formats are allowed to store data, including CSV, Excel, JSON as well as database files.

6.2.2. Data Metadata

The *metadata.json* (in Figure 11) file contains all metadata that are relevant to interpret the data. The metadata schema provides an annotation specifically for the model class data (see Section 4.2.2 for further details).

6.3. Data Visualization and Data Transformation

To visualize data, the user can provide a script that specifies which data are visualized and in which way.

6.3.1. Visualization Results

The visualized data can be stored as a figure and/or in a file (*viz1_res.r* and *plot.png* in Figure 11). See Section 4.3.2 for details.

6.3.2. Data Visualization and Data Transformation Scripts

Visualization scripts create plots that represent data (*visualization_1.r* in Figure 11). It is possible to provide also optional data transformation scripts that pre-process the data before execution of the visualization script (*dataTransformation.r* in Figure 11). Data transformation might include the extraction of a data subsets, unit transformation, or even scenarios that allow the selection between different data visualisation choices. Both, the visualization and the transformation script can be stored as described in Sections 4.2.1 and 4.3.3.

6.3.3. Required Software and Packages

The FSKX file contains detailed information about the software packages required for data visualization and data transformation (*package.json* in Figure 11).

As elucidated in Section 4.2.4, the mandatory metadata field “Language written in” specifies the required software and the version of this software and a JSON file has to specify the used packages.

6.4. Transformation Related Files

To store transformation related files the same mechanisms as described for model-related simulation scenarios are applied (see Section 4.3.1).

7. Example

In this chapter, we present examples for each of the above mentioned model types, namely single model (Chapter 4) and data file (Chapter 6). All examples are fictive and were created using the KNIME extension FSK-Lab (see de Alba Aparicio et al. (2018) and https://foodrisklabs.bfr.bund.de/fsk-lab/ for details).

7.1. Minimal Example for a Single Model

A minimal, fictive example for a single model is provided. The presented example is a dose-response (DR) model, i.e. it describes the relationship between the magnitude of exposure (dose) to a hazard and the severity and/or frequency of associated adverse effects (response) (Haberbeck et al., 2018). The presented example it is entitled “ExpDR” and can be found in the Section “Example dose-response model” on https://foodrisklabs.bfr.bund.de/fskx-food-safety-knowledge-exchange-format/).

Figure 12 presents the folder structure of the FSKX-model (see Figure 2 for the general structure and Chapter 4 for details). Model metadata, including the input and output parameters, are listed in the *metadata.json* (Figure 13). Using the input parameters, the model presented in *model.r* is executed. The simulation results are visualized using the file *visualization.r* and saved in *plot.png* (Figure 14).

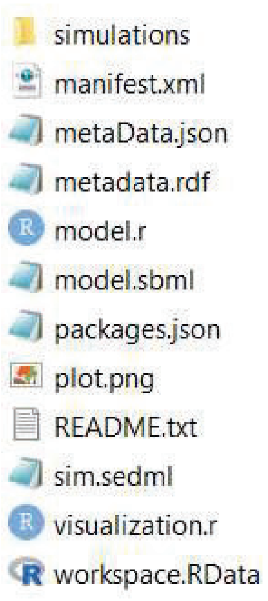

simulations
manifest.xml
metaData.json
metadata.rdf
model.r
model.sbml
packages.json
plot.png
README.txt
sim.sedml
visualization.r
workspace.RData

Figure 12 Folder structure of the example single model entiteled "ExpDR".

```
{
  "modelType": "genericModel",
  "generalInformation": {
    "name": "ExampleDoseResponseModel",
    "source": "EXPERT KNOWLEDGE ELICITATION AND EXPERT OPINIONS",
    "identifier": "ExpDRModel",
    "author": [
      {
        "familyName": "Sundermann",
        "givenName": "Esther",
        "email": "esther-maria.sundermann@bfr.bund.de",
        "streetAddress": "Max-Dohrn-Str. 8-10",
        "country": "Germany",
        "organization": "German Federal Institute for Risk Assessment"
      }
    ],
    "creator": [
      {
        "familyName": "Sundermann",
        "givenName": "Esther",
        "email": "esther-maria.sundermann@bfr.bund.de",
        "streetAddress": "Max-Dohrn-Str. 8-10",
        "country": "Germany",
        "organization": "German Federal Institute for Risk Assessment"
      }
    ],
    "creationDate": [
      2021,
      1,
      19
    ],
    "modificationDate": [
      [
        2021,
        1,
        29
      ]
    ],
    "rights": "Creative Commons Attribution 4.0",
    "availability": "Open access",
    "url": "",
    "format": ".fskx",
    "reference": [
      {
        "isReferenceDescription": false,
        "pmid": "",
        "doi": "https://doi.org/10.1016/j.mran.2018.06.001",
        "authorList": "Haberbeck, L. U., Plaza-Rodríguez, C., Desvignes, V., Dalgaard, P., Sanaa,
M., Guillier, L., ... & Filter, M.",
        "title": "Harmonized terms, concepts and metadata for microbiological risk assessment
models: The basis for knowledge integration and exchange",
        "journal": "Microbial Risk Analysis",
        "volume": "10",
        "issue": "",
        "status": "",
        "website": "",
        "comment": "",
        "abstract": ""
      }
    ],
    "language": "English",
    "software": "R",
    "languageWrittenIn": "R 3",
    "modelCategory": {
      "modelClass": "(Data)",
      "modelSubClass": [],
      "modelClassComment": "",
      "basicProcess": []
    },
    "status": "Uncurated",
    "objective": "The model is a fictive and minimal example of a dose-response model.",
    "description": "The model is a fictive and minimal example of a dose-response model. A
sigmoid function predicts a fictive effects of contamination on human health."
  },
  "scope": {
    "product": [
      {
        "name": "Any",
        "description": "",
        "unit": "[]",
        "method": [],
        "packaging": [],
        "treatment": [],
        "originCountry": "",
        "originArea": "",
        "fisheriesArea": ""
      }
    ],
    "hazard": [
      {
        "type": "Microorganisms",
        "name": "A fictive hazard",
        "description": "",
        "unit": "CFU",
        "adverseEffect": "",
        "sourceOfContamination": "",
        "benchmarkDose": "",
        "maximumResidueLimit": "",
        "noObservedAdverseAffectLevel": "",
        "lowestObservedAdverseAffectLevel": "",
        "acceptableOperatorsExposureLevel": "",
        "acuteReferenceDose": "",
        "acceptableDailyIntake": "",
        "indSum": ""
      }
    ],
    "populationGroup": [],
    "generalComment": "This is a fictive and minimal example that aims at demonstrating the
application of FSKX format.",
    "temporalInformation": "",
    "spatialInformation": []
  },
  "dataBackground": {
    "study": {
      "identifier": "",
      "title": "",
      "description": "",
      "designType": "",
      "assayMeasurementType": "",
      "assayTechnologyType": "",
      "assayTechnologyPlatform": "",
      "accreditationProcedureForTheAssayTechnology": "",
      "protocolName": "",
      "protocolDescription": "",
      "protocolURI": "",
      "protocolVersion": "",
      "protocolParametersName": "",
      "protocolComponentsName": "",
      "protocolComponentsType": ""
    },
    "studySample": [],
    "dietaryAssessmentMethod": [],
    "laboratory": [],
    "assay": []
  },
  "modelMath": {
    "parameter": [
      {
        "id": "response",
        "classification": "OUTPUT",
        "name": "response",
        "description": "A fictive probability of illness",
        "unit": "[Probability]",
        "unitCategory": "",
        "dataType": "VECTOROFNUMBERS",
        "source": "",
        "subject": "",
        "distribution": "",
        "value": "",
        "variabilitySubject": "",
        "minValue": "",
        "maxValue": "",
        "error": ""
      },
      {
        "id": "doseValue",
        "classification": "INPUT",
        "name": "doseValue",
        "description": "The considered dose",
        "unit": "CFU",
        "unitCategory": "",
        "dataType": "VECTOROFNUMBERS",
        "source": "",
        "subject": "",
        "distribution": "",
        "value": "10**rnorm(1000, -1, 1.5)",
        "variabilitySubject": "",
        "minValue": "0",
        "maxValue": "",
        "error": ""
      }
    ],
    "qualityMeasures": [
      {}s
    ],
    "modelEquation": [],
    "fittingProcedure": "Fictive function",
    "exposure": [],
    "event": []
  }
}
```
Figure 13 The file *metaData.json* of the single model example "ExpDR".

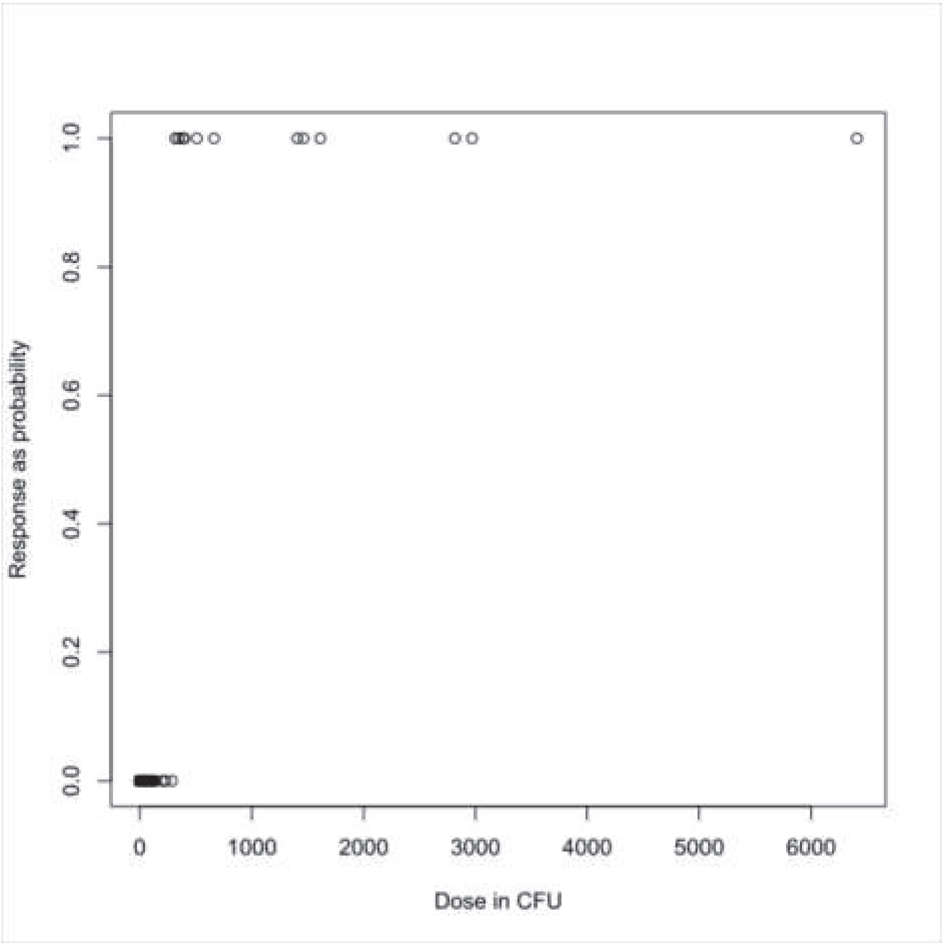
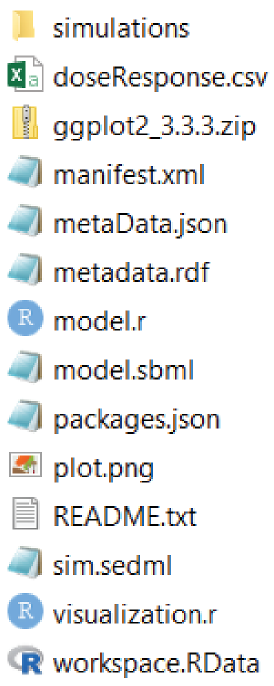


Figure 14 Simulation results of the example single model "ExpDR".

7.2. Minimal Example for a Data Set

A minimal, fictive example for a single model of the subclass "(data)" is presented. The example stores fictive data that describes the dose-response relationship; it is entitled "ExpData" (https://foodrisklabs.bfr.bund.de/fskx-food-safety-knowledge-exchange-format/).

Figure 15 presents the folder structure of the FSKX-model (see Figure 11 for the general structure and Chapter 6 for the details). Model metadata including the input and output parameter are listed in the *metadata.json* (Figure 16). Based on the input parameter *DataFileName*, the data from the *doseResponse.csv* are read into *model.r*. The *visualization.r* plots the dose-response curves for different strains of a fictive pathogen. The result is saved in *plot.png* (Figure 17).



simulations
doseResponse.csv
ggplot2_3.3.3.zip
manifest.xml
metaData.json
metadata.rdf
model.r
model.sbml
packages.json
plot.png
README.txt
sim.sedml
visualization.r
workspace.RData

Figure 15 Folder structure of the example data file entiteled "ExpData".

```
{
  "modelType": "genericModel",
  "generalInformation": {
    "name": "ExpData",
    "source": "EXPERT KNOWLEDGE ELICITATION AND EXPERT OPINIONS",
    "identifier": "ExpData",
    "author": [
      {
        "title": "",
        "familyName": "Sundermann",
        "givenName": "Esther M.",
        "email": "esther-maria.sundermann@bfr.bund.de",
        "telephone": "",
        "streetAddress": "",
        "country": "",
        "zipCode": "",
        "region": "",
        "timeZone": "",
        "gender": "",
        "note": "",
        "organization": "German Federal Institute for Risk Assessment "
      }
    ],
    "creator": [
      {
        "title": "",
        "familyName": "Sundermann",
        "givenName": "Esther M.",
        "email": "esther-maria.sundermann@bfr.bund.de",
        "telephone": "",
        "streetAddress": "",
        "country": "",
        "zipCode": "",
        "region": "",
        "timeZone": "",
        "gender": "",
        "note": "",
        "organization": "German Federal Institute for Risk Assessment "
      }
    ],
    "creationDate": [
      2021,
      2,
      9
    ],
    "modificationDate": [],
    "rights": "Academic Free License 3.0",
    "availability": "Open access",
    "url": "",
    "format": ".FSKX",
    "reference": [
      {
        "isReferenceDescription": true,
        "date": [
          2018,
          1,
          1
        ],
        "pmid": "",
        "doi": "https://doi.org/10.1016/j.mran.2018.06.001",
        "authorList": "Haberbeck, L. U., Plaza-Rodríguez, C., Desvignes, V., Dalgaard, P., Sanaa,
M., Guillier, L., ... & Filter, M. ",
        "title": "Harmonized terms, concepts and metadata for microbiological risk assessment
models: The basis for knowledge integration and exchange",
        "journal": "Microbial Risk Analysis",
        "volume": "10",
        "issue": "",
        "status": "",
        "website": "",
        "comment": "",
        "abstract": ""
      }
    ],
    "language": "English",
    "software": "FSK-Lab",
    "languageWrittenIn": "R 3",
    "modelCategory": {
      "modelClass": "(Data)",
      "modelSubClass": [],
      "modelClassComment": "",
      "basicProcess": []
    },
    "status": "Uncurated",
    "objective": "Visualization of fictive dose-response curves for two strains of one pathogen
species. ",
    "description": "Fictive dose-response curves for two strains of one pathogen species. "
  },
  "scope": {
    "product": [
      {
        "name": "Any",
        "description": "",
        "unit": "[]",
        "method": [],
        "packaging": [],
        "treatment": [],
        "originCountry": "",
        "originArea": "",
        "fisheriesArea": ""
      }
    ],
    "hazard": [
      {
        "type": "Microorganisms",
        "name": "Fictive hazard",
        "description": "",
        "unit": "CFU",
        "adverseEffect": "",
        "sourceOfContamination": "",
        "benchmarkDose": "",
        "maximumResidueLimit": "",
        "noObservedAdverseAffectLevel": "",
        "lowestObservedAdverseAffectLevel": "",
        "acceptableOperatorsExposureLevel": "",
        "acuteReferenceDose": "",
        "acceptableDailyIntake": "",
        "indSum": ""
      }
    ],
    "populationGroup": [],
    "generalComment": "",
    "temporalInformation": "",
    "spatialInformation": []
  },
  "dataBackground": {
    "study": {
      "identifier": "",
      "title": "",
      "description": "",
      "designType": "",
      "assayMeasurementType": "",
      "assayTechnologyType": "",
      "assayTechnologyPlatform": "",
      "accreditationProcedureForTheAssayTechnology": "",
      "protocolName": "",
      "protocolDescription": "",
      "protocolURI": "",
      "protocolVersion": "",
      "protocolParametersName": "",
      "protocolComponentsName": "",
      "protocolComponentsType": ""
    },
    "studySample": [],
    "dietaryAssessmentMethod": [],
    "laboratory": [],
    "assay": []
  },
  "modelMath": {
    "parameter": [
      {
        "id": "DataFileName",
        "classification": "INPUT",
        "name": "DataFileName",
        "description": "Name of the file that contains the information about the fictive dose-
response curves of various strains.",
        "unit": "[]",
        "unitCategory": "",
        "dataType": "STRING",
        "source": "",
        "subject": "",
        "distribution": "",
        "value": "'doseResponse.csv'",
        "variabilitySubject": "",
        "minValue": "",
        "maxValue": "",
        "error": ""
      },
      {
        "id": "dataDR",
        "classification": "OUTPUT",
        "name": "dataDR",
        "description": "A dataframe that includes the dose (in CFU) and the response (probability
of infection in %) for various strains.",
        "unit": "[]",
        "unitCategory": "",
        "dataType": "MATRIXOFNUMBERS",
        "source": "",
        "subject": "",
        "distribution": "",
        "value": "",
        "variabilitySubject": "",
        "minValue": "",
        "maxValue": "",
        "error": ""
      }
    ],
    "qualityMeasures": [],
    "modelEquation": [],
    "fittingProcedure": "",
    "exposure": [],
    "event": []
  }
}
```

Figure 16 The file m*etaData.json* of the data file example "ExpData".

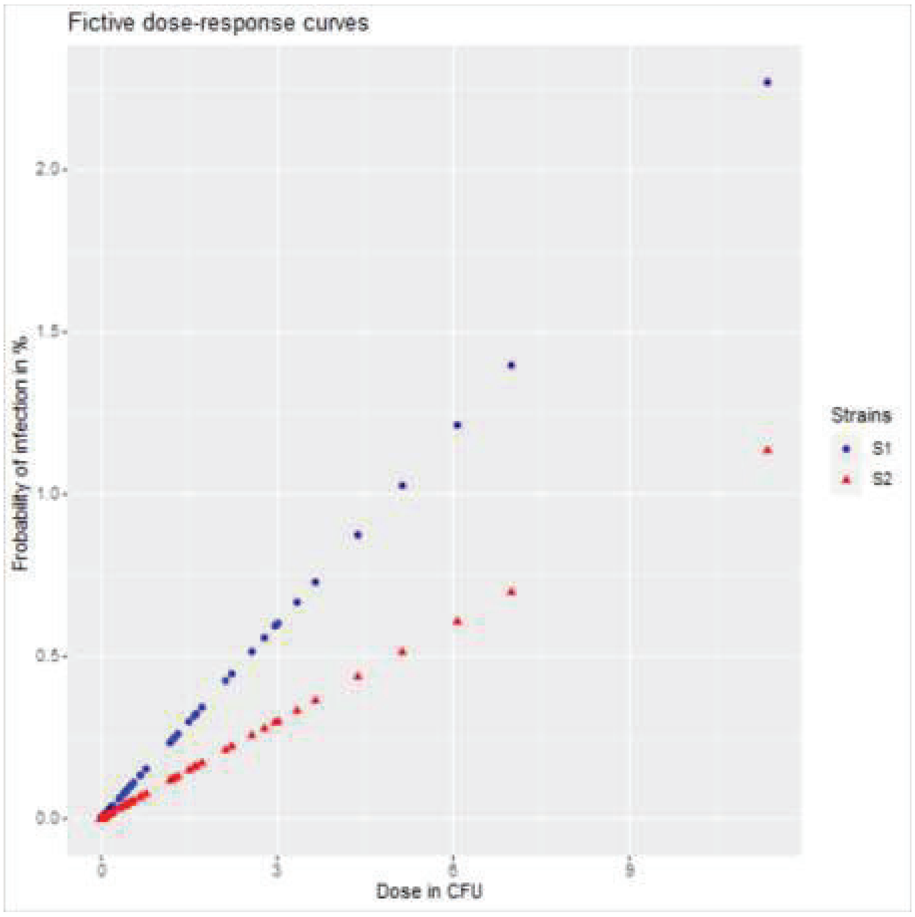


Figure 17 Simulation results of the example data file "ExpData".

8. Supplementary Information

Supplementary Information associated with this article can be found here: https://foodrisklabs.bfr.bund.de/fskx-food-safety-knowledge-exchange-format/

9. Funding

This work was supported by the EFSA-BfR Framework Partnership Agreement (FPA) GP/EFSA/AMU/2016/01, Specific Agreement Number 3 (SA3) “Establishment of a prototypic QMRA food and feed safety model repository” and by the AGINFRA PLUS project funded by the European Commission's Horizon 2020 research and innovation programme under grant agreement No 731001. EMS is funded by the JIP MATRIX within the One Health EJP. One Health EJP has received funding from the European Union Horizon 2020 research and innovation program under grant agreement No 773830.

10. References

Bergmann, F. T., Adams, R., Moodie, S., Cooper, J., Glont, M., Golebiewski, M., . . . Le Novere, N. (2014). COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics, 15*(1), 369. doi:10.1186/s12859-014-0369-z

Codex Alimentarius Commission. (1999). Principles and guidelines for the conduct of microbiological risk assessment. Joint FAO/WHO Food Standards Programme. *CAC/GL-30, Rome*.

Cyganiak, R., Wood, D., Lanthaler, M., Klyne, G., Carroll, J. J., & McBride, B. (2014). RDF 1.1 concepts and abstract syntax. *W3C recommendation, 25*(02).

de Alba Aparicio, M., Buschhardt, T., Swaid, A., Valentin, L., Mesa-Varona, O., Günther, T., . . . Filter, M. (2018). FSK-Lab – An open source food safety model integration tool. *Microbial Risk Analysis, 10*, 13-19. doi:10.1016/j.mran.2018.09.001

FAO/WHO. (2016). Codex Alimentarius commission. *Procedural Manual, 25th ed, Rome*.

Filter, M., Heise, A., Thöns, C., Perez-Rodriguez, F., Cid García, M. Á., & de Alba Aparicio, M. (2016). Predictive modelling in food markup language (PMF-ML). doi:10.13140/RG.2.1.1086.2488

Filter, M., Sundermann, E. M., Mesa-Varona, O., Buschhardt, T., Lopez de Abechuco, E., & Georgiadis, M. (2021). Minimum Information Required to Annotate Food Safety Risk Assessment Models (MIRARAM). *Food Research International, 139*, 109952. doi:https://doi.org/10.1016/j.foodres.2020.109952

Freed, N., & Borenstein, N. (1996). Multipurpose internet mail extensions (MIME) part two: Media types. *rfc 2046, November*.

Haberbeck, L. U., Plaza-Rodriguez, C., Desvignes, V., Dalgaard, P., Sanaa, M., Guillier, L., . . . Filter, M. (2018). Harmonized terms, concepts and metadata for microbiological risk assessment models: The basis for knowledge integration and exchange. *Microbial Risk Analysis, 10*, 3-12. doi:10.1016/j.mran.2018.06.001

Hucka, M., Bergmann, F. T., Hoops, S., Keating, S. M., Sahle, S., Schaff, J. C., . . . Wilkinson, D. J. (2015). The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core. *J Integr Bioinform, 12*(2), 266. doi:10.2390/biecoll-jib-2015-266

ISO (International Organisation for Standardisation). (2004). Information technology – Metadata registries (MDR) Part 1: Fr ISO/IEC 11179-1.

Lunn, D., Spiegelhalter, D., Thomas, A., & Best, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine, 28*(25), 3049-3067. doi:https://doi.org/10.1002/sim.3680

Neal, T. (2021). Overview. Retrieved from http://www.openbugs.net/w/FrontPage

R Core Team. (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. *URL:* http://www.R-project.org.

Waltemath, D., Bergmann, F., Adams, R., & Le Novere, N. (2011). Simulation Experiment Description Markup Language (SED-ML) : Level 1 Version 1. *Nature Precedings*. doi:10.1038/npre.2011.5846.1

Whiting, R. C., Buchanan, R.L.,. (1993). A classification of models in predictive microbiology - a reply to K.R. Davey. *Food Microbiology, 10*, 175–177.

Wikimedia Foundation Inc. (2019). Computer simulation. *URL: https://en.wikipedia.org/wiki/Computer_simulation*.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., . . . Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data, 3*, 160018. doi:10.1038/sdata.2016.18