FAIR Scientific Knowledge eXchange (FSKX) Format Core Specification v3.3

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

Contact:

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

This document defines the normative core specification of the FAIR Scientific Knowledge eXchange (FSKX) format, version 3.3. It is structurally aligned with the OMEX / COMBINE Archive specification but specialized for the exchange of food safety and risk assessment models, simulations, and datasets.

Contents

1 Document Conventions

FSKX follows the editorial and typographical conventions of the OMEX (COMBINE Archive) specification, adapted to cover script-based models, datasets, and FSKX-specific metadata.

Class names: Names of ordinary (concrete) classes begin with a capital letter and are written in a bold, sans-serif style.

Abstract classes: Abstract classes serve as parents of other classes and are written in bold, italic, sans-serif style.

Attributes, data types, and tokens: Attributes of classes, XML and JSON element names, literal XML, and similar tokens are written in monospace.

File names: Concrete file names that must be used exactly as specified are written in monospace, e.g., manifest.xml.

Compliance keywords: The terms MUST, SHALL, SHOULD, RECOMMENDED, MAY, and OPTIONAL are used in the sense of RFC 2119. MUST/SHALL are mandatory requirements; SHOULD/RECOMMENDED are best practices; MAY/OPTIONAL denote permitted features.

Tables: Tables summarize normative constraints, especially lists of required files, controlled vocabularies, and supported data types. Unless noted otherwise, each row expresses a requirement on conformant archives.

2 Background

2.1 Motivation

Mathematical models and data analysis workflows are central for food safety risk assessment, predictive microbiology, and related domains. To make such knowledge FAIR (findable, accessible, interoperable, and reusable), models, simulation experiments, and associated data must be shared in a software-independent and self-contained way.

The FSKX format builds on the OMEX / COMBINE Archive concept (1) of using a ZIP container with an explicit manifest and machine-readable metadata, and extends it for script-based models (e.g., R, Python) and data-centric workflows. FSKX introduces a JSON-based metadata layer tailored to food safety and risk assessment models and adds conventions for simulation settings and datasets.

2.2 History

FSKX evolved from the Predictive Modelling in Food Markup Language (PMF-ML) (2), which focused on SBML-encoded models. FSKX generalized PMF-ML towards script-based models and more flexible container structures, while integrating community-developed metadata harmonization efforts such as MIRARAM (3) and the Generic Metadata Schema (adapted from

(4); see also RAKIP Harmonization Resources). Version 3.3 consolidates experiences from early adopters and aligns the structure of this text more closely with OMEX.

2.3 Relationship to OMEX and related standards

FSKX is a specialization of OMEX. Every FSKX archive is an OMEX-compliant COMBINE archive with additional conventions on the content files. FSKX reuses the ZIP-based container, the manifest.xml file syntax, the use of Internet Media Types (5) and Identifiers.org URIs (6), and the recommended RDF-based metadata approach with Dublin Core and vCard terms (7). It adds FSKX-specific JSON metadata (metaData.json), conventions for script entry points, explicit metadata for dependencies (packages.json), and SED-ML (8) for simulation scenarios.

3 FSKX Archive Format

3.1 Archive container

An FSKX archive is encoded using the Open Modeling EXchange format (OMEX). The archive is a ZIP file containing one manifest.xml file at the root and any number of additional files. The archive SHOULD use the extension ".fskx". Paths referenced in manifest.xml MUST match the internal paths used inside the ZIP container.

3.2 File extensions

FSKX archives SHOULD use the extension ".fskx" to indicate compliance with this specification. When distributed in generic OMEX repositories, additional extensions such as ".omex", ".sedx", or ".sbex" MAY be used, but FSKX-aware tools are expected to recognize ".fskx" as the primary extension.

3.3 Required and optional content

An FSKX archive contains container-level files, model or data level files, and auxiliary files. The following table summarizes which files are required for a valid FSKX archive.

File	Location	Requirement Level	Purpose
manifest.xml	Root	MUST	Lists all files in the archive and their media types.
metadata.rdf	Root	MUST	Declares FSKX-specific roles for files via dc:type.
metaData.json	Root	MUST	JSON metadata describing the model or dataset.

packages.json	Root	SHOULD	Describes the programming language and packages required.
README.txt	Root	MUST	Human-readable instructions and contextual information.
Model script	Root	MUST for model archives	Entry-point script implementing the model logic.
Data file(s)	Anywhere	MUST for data archives	One or more files containing the dataset(s).
sim.sedml	Root	MUST	SED-ML files describing simulation or transformation scenarios.
model.sbml	Anywhere	RECOMMENDED for models	SBML representation of default parameters.

Additional files such as visualization scripts, plots, serialized simulation results, supplementary PDFs, or workspace files MAY be included and are referenced via manifest.xml.

3.4 Namespace URIs and versioning

To uniquely identify FSKX-specific document types and support versioning, the following namespace URIs are proposed (although not currently established):

- FSKX archive namespace: "http://identifiers.org/fskx/archive"
- FSK-SED-ML namespace: "http://identifiers.org/fskx/fsk-sedml"

The version of the FSKX format used by an archive SHALL be declared in metadata.rdf using the Dublin Core conformsTo property, for example:

```
<rdf:Description rdf:about=".">
  <dcterms:conformsTo>FSKX-3.3</dcterms:conformsTo>
</rdf:Description>
```

3.5 Primitive data types

FSKX reuses XML Schema 1.0 data types for XML-based artifacts (manifest.xml, metadata.rdf, SED-ML). In particular, string and boolean are heavily used in manifest.xml. JSON-based artifacts (metaData.json, packages.json, result files) follow JSON Schema typing and use controlled vocabularies where needed.

3.6 The manifest.xml file

The manifest.xml file is identical in syntax and semantics to the OMEX manifest. It SHALL contain a single OmexManifest element in the OMEX manifest namespace and one Content child per file in the archive. At least one Content element MUST describe the archive itself with location=".".

Example manifest.xml for a simple FSKX model archive:

```
<?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="metadata.rdf"
           format="http://identifiers.org/combine.specifications/omex-metadata"/>
  <content location="metaData.json"
           format="https://www.iana.org/assignments/media-types/application/json"/>
  <content location="packages.json"
           format="https://www.iana.org/assignments/media-types/application/json"/>
  <content location="model.R" master="true"
           format="http://purl.org/NET/mediatypes/application/r"/>
  <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="README.txt"
           format="http://purl.org/NET/mediatypes/text-xplain"/>
</omexManifest>
```

3.6.1 The Content class and master attribute

Each Content element declares a file contained in the archive. The location attribute is a relative path from the root of the archive. The format attribute encodes the media type or COMBINE specification URI. FSKX archives MAY set the optional master attribute to "true" for a single Content element, typically pointing to the primary model script (e.g. model.py) or, for data-only archives, to a main visualization script.

4 Metadata Specification

4.1 RDF metadata (metadata.rdf)

The metadata.rdf file provides machine-readable information about the files in the archive, including their FSKX-specific roles. FSKX adopts the OMEX recommendation and uses RDF 1.1, Dublin Core Terms, and vCard RDF terms, plus FSKX-specific dc:type values.

dc:type value	Description	Requirement Level
modelScript	Main entry-point script for executing the model logic.	MUST for model archives
annotation	JSON file containing model or data metadata (metadata.json).	OPTIONAL (will become MUST in later versions of FSKX)
dependencies	JSON file describing required language and packages (packages.json).	SHOULD
readme	Human-readable README file.	MUST
visualizationScript	Script that produces plots or reports.	OPTIONAL
workspace	Workspace or binary file with stored results.	OPTIONAL
output	File containing simulation or transformation results.	OPTIONAL

Example metadata.rdf for a single-model archive:

```
<?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:dc="http://purl.org/dc/elements/1.1/"
         xmlns:vCard="http://www.w3.org/2006/vcard/ns#">
  <rdf:Description rdf:about="."><dcterms:conformsTo>FSKX-3.3</dcterms:conformsTo>          </rdf:Description>
  <rdf:Description rdf:about="/model.R">         <dc:type>modelScript</dc:type>             </rdf:Description>
  <rdf:Description rdf:about="/visualization.R"> <dc:type>visualizationScript</dc:type>     </rdf:Description>
  <rdf:Description rdf:about="/metaData.json">   <dc:type>annotation</dc:type>              </rdf:Description>
  <rdf:Description rdf:about="/packages.json">   <dc:type>dependencies</dc:type>.          </rdf:Description>
  <rdf:Description rdf:about="/README.txt">      <dc:type>readme</dc:type>                  </rdf:Description>
</rdf:RDF>
```

4.2 JSON metadata (metaData.json)

The metadata.json file contains the core metadata describing the model or dataset. It SHALL be valid JSON and SHALL conform to the community-maintained Generic Metadata Schema (https://github.com/RakipInitiative/Model-Metadata-Schema/blob/main/genericModelMetadataSchema_104.json). At minimum it MUST contain the sections generalInformation, scope, and modelMath (for models) or dataBackground (for data-only archives). See MIRARAM guidelines (3).

Example (simplified) metaData.json for a generic model:


```
{
  "modelType": "genericModel",
  "generalInformation": {
    "identifier": "fdc00115-976a-41b4-9291-e37cf8498781",
    "languageWrittenIn": "R 4.4",
    "modificationDate": [  [  2026,2,12 ] ],
    "creationDate": [ 2026,2,12],
    "description": "Example Dose Model for FSKX Specification Document v3.3",
    "creator": [{
        "email": "example@mail.com",
        "familyName": "Doe",
        "givenName": "John"
      }],
    "reference": [{
        "title": "An Example Dose-Response Model for Fictitious Pathogen",
        "doi": "10.12345/doi.123456",
        "isReferenceDescription": true
      }],
    "name": "Example Dose Response Model",
    "rights": "Creative Commons Attribution 4.0"
  },
  "modelMath": {
    "parameter": [{
        "id": "doseValue",
        "classification": "INPUT",
        "name": "dose value",
        "unit": "CFU",
        "dataType": "VECTOROFNUMBERS",
        "value": "10^(seq(-2, 4, length.out = 100))",
        "description": "Dose values representing the amount of pathogen exposure"
      },{
        "id": "response",
        "classification": "OUTPUT",
        "name": "response",
        "unit": "[Probability]",
        "dataType": "VECTOROFNUMBERS",
        "description": "Probability of illness for each dose value"
      }]},
  "scope": {
    "hazard": [ {
        "name": "Fictitious pathogen",
        "type": "Microorganism",
        "unit": "CFU"
      }]},
  "dataBackground": {}
}
```

4.3 Parameter specification and controlled vocabularies

The parameter objects within modelMath describe inputs, constants, and outputs of the model. FSKX enforces controlled vocabularies for classification (INPUT, CONSTANT, OUTPUT) and for dataType (NUMBER, STRING, FILE, VECTOROFNUMBERS, MATRIXOFNUMBERS, OBJECT, etc.). Tools that support FSKX (e.g. FSK-Lab, https://foodrisklabs.bfr.bund.de/fsk-lab/) MUST support at least the types listed in the Generic Metadata Schema.

Parameters of type STRING or FILE SHOULD be escaped.

Example: INPUT parameter "country" has the value "Germany", the value for this parameter in modelMath SHOULD be written as "\"Germany\"". A FILE parameter (INPUT) in this context has the value of the name of its corresponding file, which is also treated as a STRING.

```
{ …,
  "modelMath": {
    "parameter": [{
        "id": "country",
        "classification": "INPUT",
        "name": "country of origin",
        "unit": "none",
        "dataType": "STRING",
        "value": "\"Germany\"",
        "description": "country where the data comes from"
      },{
        "id": "response_table",
        "classification": "OUTPUT",
        "name": "table file response",
        "unit": "[]",
        "dataType": "FILE",
        "description": "output data stored in \"response_table.csv\""
      }]},
  …
}
```

In future versions of FSKX, escaping STRING or FILE parameters will no longer be required.

5 Simulation Settings and SED-ML

FSKX uses SED-ML (8) to capture simulation and transformation settings in a software-independent way. SED-ML Level 1 allows referencing script-based models. An extension "FSK-SED-ML"(https://foodrisklabs.bfr.bund.de/wp-content/uploads/fsk/Supplement_FSK_guidance_document_restructured_V3_1.pdf) was developed in the past but is not currently used in this specification. Future versions of FSKX might revise FSK-SED-ML for adoption, to allow additional annotations for food safety scenarios.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<sedML xmlns="http://sed-ml.org/" level="1" version="1">
  <annotation>
    <SelectedSimulation>0</SelectedSimulation>
  </annotation>
  <listOfModels>
    <model id="defaultSimulation" language="https://iana.org/assignments/mediatypes/text/x-r" source="model.r">
      <listOfChanges>
        <changeAttribute target="doseValue" newValue="10^(seq(-2, 4, length.out = 100))"/>
      </listOfChanges>
    </model>
  </listOfModels>
  <listOfSimulations>
    <steadyState id="steadyState">
      <annotation>
        <sourceScript language="https://iana.org/assignments/mediatypes/text/x-r" src="./param.r"/>
      </annotation>
      <algorithm kisaoID=" "/>
    </steadyState>
  </listOfSimulations>
  <listOfTasks>
    <task id="task_defaultSimulation" modelReference="defaultSimulation" simulationReference="steadyState"/>
  </listOfTasks>
  <listOfDataGenerators>
    <dataGenerator id="response" name="response">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
        <ci> response </ci>
      </math>
      <listOfVariables>
        <variable    id="response"    symbol="urn:sedml:symbol:response"    taskReference="task_defaultSimulation"
modelReference="defaultSimulation"/>
      </listOfVariables>
    </dataGenerator>
  </listOfDataGenerators>
  <listOfOutputs>
    <plot2D id="plot1">
      <annotation>
        <sourceScript language="https://iana.org/assignments/mediatypes/text/x-r" src="./visualization.r"/>
      </annotation>
    </plot2D>
  </listOfOutputs>
</sedML>
```

In practice, SED-ML files contain one or more scenarios, each with its own parameterization, and are used to drive the execution of the main model script.

Similar to metaData.json in section 4.3, parameters with values of type STRING or FILE SHOULD be escaped, using the HTML entity for double quotation marks “"”.

Example: An INPUT parameter “country” has the value “Germany”:

```
<listOfModels>
    <model     id="defaultSimulation"     name=""     language="https://iana.org/assignments/mediatypes/text/x-py"
source="./model.py">
```

```
      <listOfChanges>
        <changeAttribute newValue="&quot;Germany&quot;" target="country" />
      </listOfChanges>
    </model>
  </listOfModels>
```

This practice will be revised in future versions of FSKX.

6. Dependencies (packages.json)

The packages.json file contains information about software dependencies of the model implementation. It SHALL be valid JSON and SHALL comply to the FSKX-Dependency schema(https://github.com/RakipInitiative/Model-Metadata-Schema/blob/main/packages-schema.json).

This JSON file MUST specify the name of the script-language used to implement the model and SHOULD specify the version of the script-language.

Furthermore, it SHOULD contain the names and versions of language specific packages (e.g. R libraries) used to run the model code.

Example:

```
{
  "Language": "R 4.4",
  "PackageList": [
    {
      "Package": "ggplot2",
      "Version": "3.5.2"
    }
  ]
}
```

7 Complete Examples

7.1 Example 1 – Single model archive

This example sketches a minimal yet complete FSKX archive implementing a single dose-response model written in R (9). The archive contains the files listed below.

```
/
├──manifest.xml
├──metadata.rdf
├──metaData.json
├──packages.json
├──README.txt
├──model.R
├──visualization.R
└── sim.sedml
```

Example model.R (simplified):

```
# model.R – Example dose-response model in R

# Input: vector 'doseValue' (CFU) defined in sim.sedml as 10^(seq(-2, 4, length.out = 100))
# Output: vector 'response' (probability of illness)

run_model <- function(doseValue) {
  # Simple sigmoid dose-response curve
  k <- 3.0
  d50 <- 1e2
  resp <- 1 / (1 + exp(-k * (log10(doseValue + 1e-9) - log10(d50))))
  return(resp)
}

response <- run_model(doseValue)
```

Example visualization.R (simplified):

```
# visualization.R – Example plot dose-response model in R

plot(doseValue, response, log = "x", type = "l",
       xlab = "Dose (CFU)", ylab = "Probability of illness")
```

Example README.txt:

```
## Dose-Response Model Example

This example demonstrates a dose-response model for a fictitious pathogen. The model calculates the probability of illness based on varying pathogen exposure doses measured in CFU (Colony Forming Units). The model takes dose values ranging from 10^-2 to 10^4 CFU as input and produces the corresponding probability of illness as output. The visualization script uses ggplot2 to create a dose-response curve with a logarithmic x-axis, making it easy to visualize the relationship across multiple orders of magnitude.
```

```
## FSK-Lab
This model is made available in the FSK-ML format, i.e. as .fskx file.To execute the model or to perform model-based predictions it is recommended to use the software FSK-Lab. FSK-Lab is an open-source extension of the open-source data analytics platform KNIME. To install FSK-Lab follow the installation instructions available at: https://foodrisklabs.bfr.bund.de/fsk-lab_de/.
```

Other files that were already exemplified: manifest.xml(see section 3.6), metadata.rdf(see section 4.1), metaData.json(see section 4.2), sim.sedml(see section 5).

7.2 Example 2 – Data archive

The second example is a pure data archive that contains a dataset and a visualization script.

```
/
├──manifest.xml
├──metadata.rdf
├──metaData.json
├──packages.json
├──README.txt
├──doseResponse.csv
└── plotDoseResponse.R
```

Example plotDoseResponse.R:

```
# plotDoseResponse.R – visualize dose-response data
library(ggplot2)

data <- read.csv("doseResponse.csv")

ggplot(data, aes(x = dose, y = response, colour = strain)) +
  geom_point() +
  scale_x_log10() +
  xlab("Dose (CFU)") +
  ylab("Probability of infection [%]") +
  theme_minimal()
```

8 Validation Requirements

To be considered a valid FSKX archive according to this core specification, an archive MUST satisfy at least the following conditions:

V1 – The file is a syntactically valid ZIP archive.

V2 – The archive contains a manifest.xml file at its root that is valid OMEX manifest XML.

V3 – All files listed in manifest.xml exist at the given relative locations.

V4 – The manifest contains a Content element with location="." and an OMEX format URI.

V5 – A metadata.rdf file exists at the root and is well-formed RDF/XML.

V6 – metaData.json is present at the root and validates against the Generic Metadata Schema.

V7 – A README.txt file is present at the root and is human-readable plain text.

Additional validation rules apply to specific model classes and may be defined by community profiles building on this core specification.

9 Future Development

Future versions of the FSKX specification may introduce explicit support for external file references, model composition metadata, cryptographic signatures for integrity and provenance, and more detailed profiles for specific modelling domains. Such extensions should remain backward compatible with the container and manifest conventions defined here.

- Proper usage of FSK-SED-ML (or revised)
- Usage of SBML Layer 3 package(https://sbml.org/documents/specifications/level-3/version-1/comp/) to support hierarchal model composition ("combined models")
- Registration of FSKX at identifier.org
- Revision of FSKX-Metadata-JSON-Schema

10 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.

This work was additionally funded by the German Federal Ministry of Agriculture, Food and Regional Identity (BMLEH) within the research project "KI- & Daten-Akzelerator (KIDA)", project number 28KIDA004.

11 References

1. Bergmann FT, Adams R, Moodie S, Cooper J, Glont M, Golebiewski M, et al. COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. BMC Bioinformatics. 2014;15(1):369.
2. Filter M, Heise A, Thöns C, Pérez-Rodríguez F, Garcia M, Aparicio M. Predictive Modelling in Food Markup Language (PMF-ML)2016.
3. Filter M, Sundermann EM, Mesa-Varona O, Buschhardt T, Lopez de Abechuco E, Georgiadis M. Minimum Information Required to Annotate Food Safety Risk Assessment Models (MIRARAM). Food Research International. 2021;139:109952.
4. Haberbeck LU, Plaza-Rodríguez C, Desvignes V, Dalgaard P, Sanaa M, Guillier L, et al. Harmonized terms, concepts and metadata for microbiological risk assessment models: The basis for knowledge integration and exchange. Microbial Risk Analysis. 2018;10:3-12.
5. Freed N, Borenstein N. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. 1996. Report No.: RFC 2046.
6. Juty N, Le Novere N, Laibe C. Identifiers.org and MIRIAM Registry: community resources to provide persistent identification. Nucleic Acids Res. 2012;40(Database issue):D580-6.
7. Cyganiak R, Wood D, Lanthaler M. RDF 1.1 Concepts and Abstract Syntax. 2014.
8. Waltemath D, Adams R, Bergmann FT, Hucka M, Kolpakov F, Miller AK, et al. Reproducible computational biology experiments with SED-ML--the Simulation Experiment Description Markup Language. BMC Syst Biol. 2011;5:198.
9. R Core Team. R: A language and environment for statistical computing Vienna, Austria: R Foundation for Statistical Computing; 2019 [Available from: http://www.R-project.org.