Risk Data Library Standard Technical Review

2023-01-31

This report summarises the findings and resulting recommendations following a technical review of the Risk Data Library Standard (RDLS) by Open Data Services Co-operative. This assessment incorporates the RDLS User Documentation and the two GFDRR GitHub repositories associated with the standard, rdl-docs and rdl-standard.

The RDLS is in an early stage of development, with a complete first version of the schema and associated documentation, development and governance processes. For a data standard at this stage of its life we found much to commend, particularly the active and longstanding involvement of a range of domain experts and stakeholders. The foundational elements of a robust and potentially long-lasting data standard are all in place and the insights and recommendations discussed in this report are all intended to help the RDLS evolve into a mature and widely adopted standard in the disaster risk management (DRM) domain.

Background

This section summarises our understanding of the domain, use cases, existing standards and initiatives. Our recommendations take into account this background work.

Domain

High quality risk data is vital for well managed disaster risk. This data helps countries reduce their risks from natural disasters and climate change. The Risk Data Library was created to make risk data easier and more effective to work with and to support risk model interoperability.

The Risk Data Library Standard (RDLS) provides a consistent structure for the following data:

- Hazard
- Exposure
- Vulnerability
- Loss
Use cases

There are three main use cases for RDLS:

- Locating and preparing risk data
- Understanding the risk data landscape
- Dataset sharing

The RDLS is intended to be of use to a wide range of audiences working in the DRM domain, but in particular disaster risk analysts and researchers preparing data for risk assessment.

Existing standards and initiatives

The risk data library is unequalled in terms of its scope, allowing for the publication of hazard, exposure, vulnerability and loss data in the one data standard.

The closest similar data standards would be Open Exposure Database (OED) and Open Results Database (ORD) curated by and for the insurance industry by OASIS, the Risk Data Open Standard (RDOS) created by RMS, and the CEDE Open Data Standard from Verisk, but these standards primarily focus on select areas of risk data (exposure and losses) and don’t cover all the same components. They have also been created with different specific audiences in mind, such as for the insurance industry. There are other standards such as PCRAFI Data collection standards for asset data which have a regional focus but where future alignment may be beneficial.

The RDLS has worked to align with commonly used standards and taxonomies, most notably the GEM Taxonomy 2.0 which the RDLS exposure schema (GED4ALL) is based upon and the UCL MOVER platform, which the vulnerability schema is built upon.

Work continues within the RDLS team to monitor, learn from and where possible align with best practice within the risk data field.

Prior reviews

The standard was reviewed by Leigh Dodds in February 2021. It is unclear which, if any, of the recommendations raised have been actioned.

Domain expert feedback

This section summarises the key pieces of user feedback we have received from a range of users.
We spoke with two members of the steering committee (Stephen Hutchings and Paul Henshaw) to gather feedback and experience regarding the RDLS. This feedback can be summarised into the following headings, the current RDLS and the future challenges and opportunities:

The current RDLS

- Lacks visibility in the risk community
- Concern that it’s another ‘data standard’ that may be short lived
- Documentation does not adequately explain how to successfully implement the standard
- There needs to be more benefit to contributors that are external to the World Bank
- Concern that RDLS may be oversimplified by combining components together
- Initial work on RDLS was good and involved multiple stakeholders
- Data included in the catalogue needs to be timely for it to be useful

The future of RDLS

- RDLS would benefit from a user guide to assist with implementation
- RLDS would benefit from examples of RDLS data.
- Conversion tools are needed between related standards.
- Documentation requires more clarity on how items are linked in the standard and which fields are mandatory.
- Thought should be given to whether the associated data catalogue needs a librarian/custodian who decides what can be published based on quality.
- Encouraging dialogue between climate change specialists and disaster risk specialists would be beneficial to the standard.

User documentation

This section documents our assessment of the RDLS user documentation against best practices for documenting open data standards and provides recommendations for improvements.

The user documentation site is a good example of open standard documentation for the stage of development RDLS is in. The key pieces of information are covered and a good deal of thought has clearly gone into considering what is needed in the documentation even if it has not yet been authored. Improvements are possible that will build upon this solid foundation.
Organization of content

Diataxis is a systematic framework for technical documentation authoring. It identifies four modes of documentation:

<table>
<thead>
<tr>
<th>what they do</th>
<th>Tutorials</th>
<th>How-to guides</th>
<th>Reference</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>introduce, educate, lead</td>
<td>guide, demonstrate</td>
<td>state, describe, inform</td>
<td>explain, clarify, discuss</td>
<td></td>
</tr>
<tr>
<td>answers the question</td>
<td>“Can you teach me to…?”</td>
<td>“How do I…?”</td>
<td>“What is…?”</td>
<td>“Why…?”</td>
</tr>
<tr>
<td>oriented to learning</td>
<td>tasks</td>
<td>information</td>
<td>understanding</td>
<td></td>
</tr>
<tr>
<td>purpose to allow the newcomer to get started</td>
<td>to show how to solve a specific problem</td>
<td>to describe the machinery</td>
<td>to explain</td>
<td></td>
</tr>
<tr>
<td>form a lesson</td>
<td>a series of steps</td>
<td>dry description</td>
<td>discursive explanation</td>
<td></td>
</tr>
<tr>
<td>analogy teaching a child how to cook</td>
<td>a recipe in a cookery book</td>
<td>a reference encyclopaedia article</td>
<td>an article on culinary social history</td>
<td></td>
</tr>
</tbody>
</table>

Clear separation of content into these sections can aid users in finding the information they require in a given situation. It can also aid the writers and maintainers of the documentation in ensuring all necessary information is covered and help minimise sources of potential inconsistencies whilst implementing updates.

We assessed the current documentation structure and content against the Diataxis framework and identified the modes currently addressed by each section.

<table>
<thead>
<tr>
<th>Section</th>
<th>Modes</th>
<th>Recommended Mode</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key concepts</td>
<td>Reference</td>
<td>Explanation</td>
<td>Currently this section just contains copies of</td>
</tr>
<tr>
<td>Section</td>
<td>Modes</td>
<td>Recommended Mode</td>
<td>Comments</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>definitions from the UNDRR terminology. This is useful information but it could be expanded to more clearly explain the context of these concepts within the data model. (example)</td>
</tr>
<tr>
<td>Core standards</td>
<td>Explanation</td>
<td></td>
<td>This section is currently missing content. Any discussion around how these core standards have been used in the creation of the RDLS would be Explanation.</td>
</tr>
<tr>
<td>Taxonomies</td>
<td>Reference and Explanation</td>
<td>The content in this section should be split into two pages: Reference and Explanation</td>
<td>The paragraphs that discuss how and why the taxonomies have been selected and developed belong in an Explanation section. The details of the RDLS Hazard Taxonomy belong in the reference section.</td>
</tr>
<tr>
<td>Data model</td>
<td>Reference and Explanation</td>
<td>Reference</td>
<td>Restructure the content in these pages: Schema attribute tables should be given in a Reference section (example). Attribute descriptions should be contained in these tables rather than discursive paragraphs. Discussion of why the schema contains certain attributes should be moved to an Explanation section.</td>
</tr>
<tr>
<td>Implementation</td>
<td>How-to guide</td>
<td>How-to guides</td>
<td>Only 1 part of this section</td>
</tr>
</tbody>
</table>
### Risk Data Library Standard Technical review report

<table>
<thead>
<tr>
<th>Section</th>
<th>Modes</th>
<th>Recommended Mode</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tutorials</td>
<td>?</td>
<td>Tutorials</td>
<td>contains significant content and it is How-to content. Suggest renaming this section to “How-to-guides”</td>
</tr>
<tr>
<td>About</td>
<td></td>
<td>Explanation</td>
<td></td>
</tr>
</tbody>
</table>

Diataxis is a [pragmatic framework](#) that recommends making improvements to documentation in an iterative fashion. Much of the current content can be retained as part of a restructuring process alongside authoring some additional sections.

<table>
<thead>
<tr>
<th>Section</th>
<th>Mode</th>
<th>Description of Content</th>
<th>RDLS existing content</th>
<th>Additional content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primer</td>
<td>Explanation</td>
<td>Introduction to the domain, key concepts and reasons for standardising data, description of how the standard was developed and why, who the intended users are</td>
<td>Introduction, Taxonomies (but link to definition of RDLS Hazard taxonomy in other section), paragraphs in Data model pages discussion why certain attributes are included</td>
<td>Core standards, History</td>
</tr>
</tbody>
</table>

<p>| Schema reference | Reference | The schemas, codelists and rules that need to be followed | Data model attribute tables and examples, RDLS Hazard |                           |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Mode</th>
<th>Description of Content</th>
<th>RDLS existing content</th>
<th>Additional content</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>to publish data</td>
<td>taxonomy</td>
<td></td>
</tr>
<tr>
<td>Guidance</td>
<td>How-to guides</td>
<td>step by step guides to implementing the data standard.</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td>Glossary</td>
<td></td>
<td>Key concepts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td>How-to guides</td>
<td></td>
<td></td>
<td>support details</td>
</tr>
</tbody>
</table>

**Recommendations**

Restructure the existing content incorporating the distinction between [normative and non-normative content](#) and following the guidelines in the above table.

Where necessary author new content including:

- How-to guides
- Tutorials
- Core standards reference
- Core standards explanation
- Help and support

**Completeness**

Documentation should contain all the relevant information to answer a user's most likely questions. In particular, the reference documentation should describe the meaning of each field and code in the data model.

The current documentation site already contains much of the content that good user documentation should cover. The structure and content of the documentation was compared to that of other open data standards including [OCDS](#), [OFDS](#) and [360Giving](#), to identify any missing content. This assessment identified a number of issues with completeness that can be divided into 2 categories:

1. Empty or incomplete sections where the presence of a page or heading indicates that this content has been identified as necessary but hasn't yet been authored, e.g. [Vulnerability examples](#), [Deploy tutorial](#), and [Core standards](#).
2. Content that a user would expect to find that is not present in any form including:
   ○ Lack of descriptions for codes used in the data model. E.g. **Frequency type** in the Hazard schema has 3 possible types, "Rate of Exceedence", "Probability of Exceedence", "Return Period". The concept of a Return Period is described in the accompanying paragraph but it is not made explicit that this is the definition of the “Return Period” type nor are descriptions of the other 2 types provided anywhere in the documentation.
   ○ Information on where to get support when implementing the standard.

### Recommendations

Complete the missing content in line with the recommendations for restructuring and separating out normative and non-normative content.

#### Normative vs non-normative content

Best practice in standards documentation clearly separates normative (prescriptive) and non-normative (informative or descriptive) content. This can be achieved through clear structuring and the use of normative key words as defined by RFC2119. Clear separation of these content types can aid the user in identifying the information they need to complete a given task, and aid the document maintenance process.

The [current documentation site](#) combines normative and non-normative content types. E.g.

- The [Data model section](#) includes definitions of the parts of the schema (normative content) alongside discussion of potential use (non-normative content) and examples (non-normative content).

The use of normative keywords is inconsistent throughout the documentation site. E.g.

- On the [taxonomies page](#) the RDLS Hazard Taxonomy and GED4All taxonomy are both marked as “recommended” which is an appropriate use of this normative keyword.

- The RDLS taxonomy is marked as “recommended” in the [Taxonomies page](#), but the Hazard Type values in this taxonomy are given in schema tables Type columns (e.g. in the [Hazard schema](#)) implying that only these values can be selected. If this is the case then this taxonomy is “required” not “recommended”.

- Numerous instances of non-normative keywords where normative keywords should be used, e.g. in the [description of GED4ALL](#) the final sentence “This is the suggested option…” should be changed to “This is the recommended option…”
**Recommendations**

Separate out normative and non-normative content as part of restructuring the documentation, making use of normative keywords for normative content and using appropriate synonyms for non-normative content.

**Consistency**

Consistency within documentation is vital. Inconsistencies create confusion and can diminish the users confidence in the standard.

Inconsistencies were assessed within the documentation itself, both in its content and structure. Inconsistencies between the documentation and the schema were assessed as a key benefit of maintaining a [Single Source of Truth](#).

**Content**

Terms and concepts should be referred to consistently throughout the documentation. Consistent terminology helps users comprehend the documentation correctly. There are multiple instances where this is not the case. E.g.

- In the [RDLS Hazard taxonomy](#):
  - The top level is initially referred to as hazard in “The RDLS Hazard Taxonomy classifies hazard phenomena as main hazard (8 categories) and hazard process (27 categories)” but is referred to in the accompanying table and all following pages as hazard type. The latter term is preferable to avoid ambiguity when the key concept of Hazard is being referred to.
  - The second level is initially referred to as hazard process, “The RDLS Hazard Taxonomy classifies hazard phenomena as main hazard (8 categories) and hazard process (27 categories)**, but the accompanying table defines it as process type. Hazard process is the term used in the standard schemas.

- On the [Core Standards page](#) reference is made to “the RDL data model” rather than “the RDLS data model”.

- The [exposure example](#) includes Geographic coverage which doesn’t feature in the exposure schema but almost matches Geo coverage in the General schema.

Attributes that refer to the same concept should be consistently named: E.g.

- **Intensity unit** in the Vulnerability attribute table has the same description as Unit of measure in the Hazard attribute table.
All fields, concepts, terms etc. should be fully defined once, and then linked back to when referenced internally. This applies to all parts of the documentation, text, tables, and examples.

- Full descriptions and definitions are missing within numerous sections. E.g.
  - RDLS Hazard taxonomy values are not described.
  - The RDLS Hazard taxonomy hazard type measure metrics table doesn’t have Metric:Unit for every Hazard type in the taxonomy. Convective Storm, Extreme Temperature, and Strong Wind are all missing.
  - “Loss” is not defined in Key Concepts.

- Information is duplicated rather than providing a link to the initial definition. E.g.
  - In the Hazard page the RDLS Hazard type taxonomy is repeated.
  - In the Loss page a section describing specific periods of time is repeated from the Hazard page.

- First instances of abbreviations are not fully defined. E.g.
  - Catastrophe risk models (cat models) which first appears in https://docs.riskdatalibrary.org/keyconcepts.html#cat-model
  - The values in the Hazard type column in the RDLS Hazard taxonomy measure metrics table.

- Few internal links are used. Key areas where a link would be useful include the schema tables where the value should be taken from part of one of the recommended taxonomies.

- One instance of a circular link was identified - “please see this here” link in Other hazard taxonomies.

Structure
Consistent structure and format helps users comprehend the documentation quickly.

The structure and format used in the schema definition tables is inconsistent:

- Schema tables do not have consistent columns. The majority have 4 columns named, “Required”, “Attribute”, “Description” and “Type”.
  - The General attributes individual resources table is missing the “Description” column.
  - The Vulnerability specifics table is missing the “Required” column.
  - The Vulnerability additional and specific tables have a “Field name” column rather than an “Attribute” column.
  - The Vulnerability additional table has an “Example” column rather than a “Type” column.
The schema tables "Type" columns are inconsistent:
  o Values given are mixtures of codelists, types and formats. These should be separated out and defined elsewhere in the documentation.
  o It’s unclear if the value in this column is a fixed required value (e.g. Format = “ext” in the General attributes individual resources schema) or if the value is merely an example. This links back to the issue of the RDLS Hazard taxonomy being “recommended” but the Hazard Types from it being listed in the Hazard, Vulnerability and Loss schema “Type” columns.
  o Potential values are not ordered consistently, e.g. Calculation method in the Hazard schema is [Simulated, Observed, Inferred] and in the Loss schema is [Inferred, Simulated, Observed]. This latter order matches the ‘enum’ order in the schema definitions.

Recommendations

  ● A number of the issues described will be addressed through applying the recommendations from the Organisation of content section.

  ● Expand the Key concepts section to include more of the repeated concepts and link to them were necessary.

  ● Where multiple terms are used for the same concept, decide on a single term.

  ● Ensure that the columns within schema tables are consistent.

Single source of truth

This section documents our assessment of the user documentation with respect to the importance, for both users and the standard maintainers, of preserving an authoritative single source of truth for RDLS.

A single source of truth (SSOT) is an authoritative primary location for the definition of a data model, including the structure, format and meaning of its fields and codes.

Using a SSOT reduces the burden of maintaining a data standard by ensuring that changes and updates need only be made in one place, and provides users with confidence that they are following the correct guidance when working with the standard.
Consistency between documentation and schema

The documentation should reflect the content in the schema. Two versions of the schema were identified, one in each of the RDL GitHub repositories. The schema in the rdl-standard repository was used to check for inconsistencies between the schema and the documentation.

- Multiple inconsistencies were identified:
  - attributes in the schema but not the documentation and vice versa,
  - attributes with different names in the schema and the documentation,
  - identification of which attributes are requirements of the standard.

Appendix 2 contains a full comparison of attributes in the documentation schema tables to fields in the JSON schema.

- Values contained in the enum property of schema definitions do not all match those given in the documentation schema table ‘Type’ column. E.g.
  - Hazard type in the documentation lists the Hazard types from the RDLS Hazard taxonomy as full names but the definition of ‘common_hazard_type’ lists abbreviated codes. The same codes are used but undefined in the RDLS Hazard taxonomy measure metrics table in the documentation.

A full list is given in Appendix 2.

Reference documentation and generation

Reference documentation provides the technical description of the standard. It is a frequently used section of the documentation and as such its accuracy is vital.

Reference documentation can be generated manually, or through the use of pre-commit scripts to reformat technical definitions in JSON schema and codelist CSV files into human-readable documentation.

The RDLS user documentation appears to have been manually authored through a number of markdown files held in the rdl-docs GitHub repository. These files were then compiled and hosted as a static website using MkDocs.

The files in the repository are not all present in the MkDocs site (usecases.md is missing). There is no documentation detailing how and when these files are maintained or what the process for this is.

The inconsistencies between the documentation and schema and the creation of some of the missing content could be addressed by switching to Sphinx for documentation generation. Open Data Services maintain a number of open source Sphinx extensions to generate documentation based on JSON Schema and CSV codelist files. Moving to Sphinx...
would reduce the risk of the documentation and schema being out of sync and reduce the maintenance burden.

<table>
<thead>
<tr>
<th>Element</th>
<th>RDLS</th>
<th>Sphinx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema reference tables</td>
<td>Hardcoded (example)</td>
<td>Generated from schema using JSON Schema Directives — sphinxcontrib-opendataservices documentation (example)</td>
</tr>
<tr>
<td>Codelist reference tables</td>
<td>Hardcoded (example)</td>
<td>Generated from codelists using Misc Directives — sphinxcontrib-opendataservices documentation (example)</td>
</tr>
<tr>
<td>Definitions</td>
<td>?</td>
<td>Quoted from schema using JSON Include Directives — sphinxcontrib-opendataservices documentation (example)</td>
</tr>
<tr>
<td>Examples</td>
<td>Hardcoded (example)</td>
<td>Generated from example files using JSON Include Directives — sphinxcontrib-opendataservices documentation and reStructuredText Directives – CSV Table (example)</td>
</tr>
</tbody>
</table>

Recommendations

- Declare a Single Source of Truth, e.g. the rdl-schema GitHub repository
- Move the documentation from MkDocs to Sphinx to take advantage of the auto-generation scripts available to create reference content from the schema json.
- Remove inconsistencies between the schema and the documentation, and
Example data

This section documents our assessment of the example data provided in the user documentation and makes recommendations for improvements touching on the fullness, format, and number of examples.

Examples of standardised data are a key element of documentation, aiding the user in understanding how the standard should be applied and how the final product should look. Examples should be complete and accurate with additional explanations where appropriate and provided for all core parts of the standard. Maintaining comprehensive examples also aids standard maintenance, acting as a check on changes or updates to the data model and schema.

The Data model section of the RDLS user documentation contains a number of examples. The majority of examples take the form of a data table with attributes from the schema in one column, example values in another and a column indicating if the attribute is a requirement of the standard. Most also include an accompanying image showing the dataset being described overlaid on a basic administrative or road map. These examples are all well chosen and will aid potential users in understanding how to apply the standard to their own datasets. Improvements are possible however.

- The examples do not include all potential attributes of the standard.
  - Some of the examples are missing attributes that are defined as “Required”, e.g. the Observed losses example.
  - Some non-required attributes are not used in any example, e.g. the Flood hazard maps for Kabul example includes all of the required attributes but not all of the optional attributes.

There should be at least one example use of each attribute in the standard.

- Some, but not all, examples include attributes from the General schema as well as the specific schema, e.g. the Exposure example includes Geographic coverage which doesn’t feature in the exposure schema but almost matches Geo coverage in the General schema.

- There are no examples provided for either the General schema or the Vulnerability schema. Examples should be provided for all sections of the standard.

- The example tables include the example data in the column “Example”. The data table is already identified as an example, therefore this column would be better renamed as “Value”.

Open Data Services Co-operative Limited is a company limited by shares, operated as a Workers’ Co-operative | Registered in England. Company number: 09506232 | Registered address: 1st Floor, Holyoake House, Hanover Street, Manchester, Greater Manchester, M60 0AS | Correspondence to contact@opendataservices.coop
● The examples are provided in a single format, as an HTML table. Providing examples in the formats in which users are expected to publish data would help users to understand what implementation looks like in practice.

- The **Implementation/Local** section describes implementing RDLS via folder and file names yet no examples are provided in the Data Model sections to demonstrate this implementation option.

- The images that accompany the examples contain at least 2 datasets, the dataset being described by the example RDLS table and an underlying map.
  - Both datasets should be cited and where possible linked to.
  - There is no clear explanation of what the dataset image is representing. They require a figure caption that explicitly links the attributes in the RDLS data to the image.

- In addition to issues already mentioned there are multiple problems with the **Exposure example**:
  - Multiple attribute names do not match the attributes named in the schema table.

<table>
<thead>
<tr>
<th>Example attribute</th>
<th>Assumed schema attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exposure category</td>
<td>Category</td>
</tr>
<tr>
<td>Taxonomy</td>
<td>Taxonomy source</td>
</tr>
<tr>
<td>Period of occupancy</td>
<td>Occupancy time</td>
</tr>
</tbody>
</table>

Open Data Services Co-operative Limited is a company limited by shares, operated as a Workers’ Co-operative | Registered in England. Company number: 09506232 | Registered address: 1st Floor, Holyoake House, Hanover Street, Manchester, Greater Manchester, M60 0AS | Correspondence to contact@opendataservices.coop
There is an attribute included that can only be found in the Hazard schema, **Unit of measure** but the example value given is not an appropriate value for the attribute defined in the Hazard schema.

There is one image for two datasets but it’s unclear how the data shown relates to each of the datasets described. Suggest splitting the image to show each dataset individually and then a combined image at the end.

- There are examples provided as prose text rather than as RDLS. E.g.
  - “For example, an analysis of earthquake frequency based on seismic observations from 1934 (occurrence time start) to 2001 (occurrence time end), for a total count of 66 years (occurrence time span).” from the Hazard schema page.

### Recommendations

- Provide complete examples for all 5 sections of the standard.
- Ensure examples are consistent with the schema definitions.
- Provide examples in each supported publication format as well as HTML tables.

### Style, readability, spelling and grammar

This section discusses the assessment of the user documentation and its style, readability, spelling and grammar.

The style and readability of user documentation can have a significant impact on the adoption of the standard and should be tailored towards the intended audience. From the use cases presented the intended audience is expected to have basic technical knowledge and ability to work with data, and will be comfortable with the terminology used in the risk data management domain. Inconsistent style, poor spelling and grammar and low readability can all lead to documentation that is hard to understand and use.

The general style of the RDLS user documentation is consistent and appropriate for the intended audience, balancing an expected knowledge of the technical domain with an expected lower familiarity with metadata standards. Restructuring the existing content as recommended in [Organization of content](#) would improve the readability of the documentation.

Creating and adopting a [Style guide](#) would help to ensure the consistency of style and readability of the documentation. Issues with the current documentation to consider covering in such a guide include:
• The removal of overly long sentences, e.g.

“The exposure schema covers a wide variety of data describing structural, infrastructural and environmental asset, population, and socio-economic descriptors, each with relevant attributes for assessing risk from multiple hazards.”

could be rewritten as:

“The exposure schema covers a wide variety of data describing:

• structural, infrastructural and environmental assets
• population descriptors
• socio-economic descriptors.

with relevant attributes for assessing risk from multiple hazards.”

Where the sentence has been restructured to separate out the list and break up the long line of text. Tools such as Hemingwayapp can be used to test the readability of sentences and paragraphs as they are being written.

• Descriptions should begin with a noun phrase, e.g. “For probabilistic scenario, the occurrence probability is expressed according to frequency type” should be “The occurrence probability is expressed according to the frequency type for probabilistic scenarios.”

• Incomplete lists should begin with “e.g.” or “for example”, e.g. “(geological, geophysical and hydrometeorological)” from the definition of Hazard in Key concepts should be “(e.g. geological, geophysical and hydrometeorological)”

• When quotations are used, e.g. in Key concepts, these should be formatted as such and clearly cited.

Considerations of accessibility should be taken into account when making decisions around colour and font choice, and image descriptions. E.g. The titles in the Examples of risk data images are almost illegible in white against a pale grey background, and no image description is provided.
Minor typos and spelling mistakes are scattered throughout the documentation, e.g.

- “eartquakes” instead of “earthquakes”.
- “a dataset represent landslide hazard” instead of “a dataset representing a landslide hazard”.

**Recommendations**

- Correct spelling and grammatical errors.
- Take into consideration accessibility concerns and readability of sentences when redrafting content.

**Schema**

This section documents our assessment of the RDLS JSON schema against best practices for authoring JSON schema and makes recommendations for improvements.

Where the user documentation provides human readable explanations and references to the data standard, the schema provides a machine readable version. This element of the data standard is at the core of creating and utilising tools for both the users and maintainers of the standard, such as validation tools and documentation automation tools. A schema that is a well written and accurate representation of the data standard can ease the use of, and increase the adoption of, the standard.

It is unclear from the current documentation and files Open Data Services had access to if the intent is for a JSON file to be a requirement of valid RDLS. Due to the existence of
**General observations**

This section documents our general observations on the validity and completeness of the schema. Where fuller consideration is warranted a dedicated section is included further on in the report.

There are two versions of the schema in the GFDRR GitHub repositories:

- rdl-standard/blob/main/specs/DDH_compliant/rdl_schema_0_1.json
- rdl-docs/blob/main/docs/schema/rdl_schema_0.1.json

These versions appear to be identical but the presence of two versions of the schema increases the risk of untracked changes through the lack of a single source of truth. This issue is further discussed in Repositories.

One version of the schema was chosen for analysis, rdl_schema_0_1.json. This file is valid JSON, however a number of general issues were identified:

- The schema makes use of anyOf to select one of the four RDM components, Hazard, Exposure, Vulnerability, or Loss. This should be changed to oneOf as no dataset will have more than one of these components. If anyOf is retained additionalProperties must be disallowed.

- There are unresolvable references. After fixing these the following data validates:

```
{
    "common": {}
}
```

Presumably, this is not intended. The solution is to either:

- Add required properties to common
- Set minProperties on common
- Add required properties to all of the objects in anyOf
- Disallow additionalProperties for all of the objects in anyOf

- Related to the above point, the use of the required property is inconsistent with some second and third level objects having no required properties.
  - e.g. anyOf.vulnerability, yet there are attributes identified in the documentation as required for this element.
Data model

The data model defines the structural and conceptual relationships between the elements that make up the data standard. It can therefore have a significant impact on the usability and extensibility of the standard.

The top-level conceptual object is a data file with spatial properties. The standard started life as an SQL database designed to hold these data files themselves (the Risk Data Library) and not just catalogue the metadata and access points. This is likely to account for some of the structural decisions seen in the JSON schema. The use cases and ambitions for the standard are however more far reaching, intending to create a standard that can be used by others to expose their data from their own host locations.

JSON format allows for multiple layers of nesting. Multiple layers of nesting can make a schema harder to read for human users, and introduce complexity when flattening the data structure in order to present it in a flat field format such as a CSV file or an SQL database. The nesting in the schema is both minimal (reflecting the standard’s origins as an SQL database) and unnecessarily complicated.

- **common** consists of two nested objects, **contribution** (of type object) and **resources** (of type array). This separation is required if multiple files are to be described by **resources**, e.g. different formats or resolutions of the same data.

- The four objects representing the four components, **hazard**, **exposure**, **vulnerability** and **loss** are nested within **anyOf**, these should be at the same level as **common**. This would ease human readability.

- It’s unclear why the nesting within three of these four is necessary, e.g.
  - **exposure** is divided into **model** and **value**
  - **vulnerability** divided into **model** and **specifics**
  - **loss** having a further nested object **model**.

If the intended user of the standard is most likely to be working with spreadsheets and relational databases it makes sense to leave the schema with as few layers of hierarchy as possible. We suggest removing this second level nesting within each of these three objects.

Nesting doesn’t have to create additional complexity, it can be used to reduce repetition and therefore simplify the schema. Such a use would be through the creation of common objects. The benefit of this is that the common objects only need to be defined once in **definitions**, and the different contexts in which they are used can be given in the
description of the field that references it. The current schema makes little and inconsistent use of modularisation. E.g. the following common objects could be created:

- A taxonomy object similar to the OCDS Classification object to contain the current taxonomy_source and taxonomy_code fields in the vulnerability.model and exposure.model objects.
- An occurrence time object containing time_start, time_end, time_span, time_year from both the hazard.event_set and loss.model schema sections.

This common object method is used in exposure.value, an object made up of two fields, val_type and val_unit. However, these same two fields appear in vulnerability.model and loss.model but not within a higher level value object.

The data model is designed to describe a single dataset. However the use cases for the standard are based around the generation of multiple datasets within single projects. A method for implementing RDLS at a project/multi-dataset level is required. A simple solution would be to include a globally unique project identifier in common and make this field a required field.

The schema makes use of the definitions property defined in JSON schema to define fields and objects that are repeated within the schema. The use of this property is good practice as it means fields are defined just once, allowing definitions to act as an internal single source of truth. However, definitions should be more fully used to define all objects and not just provide codelists. This would help with the schema’s human readability by making the top level objects clearer and reduce repetition. It would also bring the standard inline with other mature standards and their validation tools.

**Recommendations**

- remove unnecessary nesting in exposure, vulnerability and loss to improve both readability and potential for flattening.
- move definitions of all objects including the higher level hazard, exposure, vulnerability and loss to definitions.
- consider creating objects for common groups of fields.

**Unresolvable references**

A number of references provided in definitions are unresolvable:
### Usability

This section documents our assessment of the schema's usability against best practices and with reference to both the user documentation and user feedback already covered.

The schema's usability should be considered from the point of view of automated tools using it, as well as the point of view of users referring to it directly.

- The assessment of the schema's machine usability is implicit in the other subsections of **Schema**.

- The primary issue with the human readable usability of the schema is the lack of consistency between the schema and the user documentation and the minimal descriptions provided for the fields and codelists. Once these issues have been addressed the schema's usability will greatly improve.

### Codelists

This subsection documents our assessment of the use of codelists within the schema, and discusses the advantages of defining these inline vs CSV.

When designing a standard it may be necessary to define a set of values that a field should (or is recommended to) use. In JSON schema this can be done in one of two ways, inline using the enum property or through the use of CSV codelists.

The use of CSV codelists has a number of advantages over the inline option. These include:
• The option to use an extension to JSON schema 0.4 developed and maintained by Open Data Services Co-operative to declare the codelist, declare if it is open (the value is only recommended to come from the list) or closed (the value should come from the list to be considered valid against the standard) and produce relevant validation messages related to the codelists. `enum` is the equivalent of a closed codelist, meaning the only acceptable value for the relevant field is one contained in the `enum` array.

• The ability to include descriptions and labels in the CSV file alongside the codes.

The schema currently uses the inline `enum` method of defining codelists alongside listing codes in field descriptions. The documentation for a number of these however describes the values given as recommended only meaning if a user uses a different (equally valid) value it will violate the standard.

**Recommendations**

- Convert the current codelists to CSV files, adding in descriptions and labels.
- Add the `codelist` properties to the schema to define if the codelists are open or closed removing the `enum` property for those fields with recommended only values.

**Titles and descriptions**

This subsection documents our assessment of the titles and descriptions of schema fields and makes recommendations for improvements based on best practice.

Titles and descriptions are the first parts of a data standard that users see. It is important therefore that they are clear, unambiguous, and consistent. The field titles and descriptions in the current version of the schema, `rdl_schema_0_1.json`, incorporate clear word choices, using simple English and in the most part only employing technical terms where it is expected that the average user would be very familiar with them. At this point in the development of a data standard this is what we would hope to see. Improvements are possible however. Based on comparison with other more mature open data standards including [OCDS](#) and [360Giving](#) the following issues were identified:

- Descriptions are given for most fields. However the majority of these are minimal and may not be easily interpretable to a non-expert user.
- The field names are not the same as the field names given in the documentation.
● Potential code values are listed in field descriptions, e.g.
  anyOf.vulnerability.model.f_relationship.
  Where a field must take a value from a specific set of codes use a codelist to specify these rather than list them in the field description.

● Some schema descriptions include format specifications, e.g.
  anyOf.hazard.event_set.time_start.description = “The time at which the modelled scenario(s) starts [ISO 8601 format]”
  If the value of the field must be provided in a specific format the format schema property should be used.

● common.resources.epsg appears to match the documented schema attribute Reference coordinate system with a Type of CRS EPSG, but it is not specified that the coordinate system used must be taken from the EPSG. Suggest either changing this to a more neutral field name, e.g. reference_coordinate_system OR specifying in the documentation and schema that EPSG codes must be used.

● There are typos in numerous field names, e.g.
  anyOf.hazard.event.occurrence_time_start should be
  anyOf.hazard.event.occurrence_time_start.

● Abbreviations and contractions in field titles should be avoided, e.g.
  anyOf.vulnerability.model.val_type would be better as
  anyOf.vulnerability.model.value_type and
  anyOf.vulnerability.model.imt would be better as
  anyOf.vulnerability.model.intensity_measure_unit

Many of these issues can all be addressed and avoided in future through reference to a style guide such as the OCDS schema style guide when reviewing and authoring field names and descriptions.

**Recommendations**

Review and rewrite field names and descriptions following a style guide such as the OCDS schema style guide.

**Object identifiers**

This subsection documents our assessment of the use of object identifiers in the schema against best practices.
Object identifiers help to ensure that separate objects within arrays remain clearly unique when the data is flattened.

The RDLS schema contains two arrays, `common.resources` and `anyOf.hazard.geo_area`. Neither of these arrays contain a unique object identifier field.

This would be a particularly important addition for the `common.resources` object that contains the details of the individual datafiles described by the rest of the RDLS. The required properties of this object, `name`, `format`, `epsg` and `url` might form a unique combination but the inclusion of a unique object identifier for each file, e.g. `id`, would remove the risk of conflating files on flattening of the data.

**Recommendations**

Add object identifiers to the `common.resources` array and consider adding them to the `anyOf.hazard.geo_area` array.

Links to external standards and taxonomies

This subsection documents our assessment of the links between the RDLS schema and external standards and taxonomies.

The RDLS schema does not explicitly link to any external standards or taxonomies.

- The documentation lists a number of suggested and recommended taxonomies, e.g. GED4ALL, but none that are required.
- ISO 8601 format is mentioned in the description of a number of time related fields but this is not enforced by the schema.
- `common_iso` appears to be a list of country codes taken from ISO 3166-1 a3 but this isn’t fully specified with both fields that reference this definition just referring to either “ISO codes(s)...” or “ISOa3 list of countries...”.

**Recommendations**

Where an external standard or taxonomy is required explicitly state this in the documentation and schema.

Technical considerations

This section documents our assessment of the technical aspects of the RDLS schema against best practices and makes recommendations for improvements.
JSON schema version and extensions

This subsection documents our assessment of the schema’s use of JSON schema versions and extensions.

JSON schema is a declarative language used to validate your own JSON documents. In the case of an open data standard this document is both the standard schema and any files declared as obeying this standard. The use of JSON schema is in the validation it enables.

The RDLS schema declares the use of the 2020-12 JSON schema, the most up-to-date version of JSON schema. However the schema does not validate against the 2020-12 JSON schema due to the use of definitions, a property that has been deprecated in this version of JSON schema in favour of defs.

The RDLS schema does not use any JSON schema extensions. Extensions are available that could improve the usability of the standard as noted in Codelists.

Recommendations

Ensure rdl_schema_0_1.json validates against the declared version of JSON schema either by updating the schema properties used, or declaring an older version of JSON schema.

Range of implementation approaches

This subsection documents our assessment of the range of implementation approaches available and possible for using the schema.

It is unclear from the existing materials in both the GitHub repositories and the documentation what implementation approaches are available.

The Hazard data package documents an implementation approach for Hazard data only that uses a slightly altered version of part of the RDLS schema.

Recommendations

Establish and document the implementation approaches that will be supported by the standard based upon user needs.
Change Management and version control

This section documents our assessment of the approach to tracking and managing changes to RDLS, and provides recommendations for improvements.

Continuous integration and automated tests

Continuous integration (CI) is a software practice that requires frequently committing code to a shared repository and triggering automated build and tests. In the context of a data standard, CI and automated tests help to ensure that changes do not invalidate the schema or cause the documentation build to fail, amongst other checks.

GitHub Actions provides a CI platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production. Workflows can run tests directly or using external testing tools and frameworks such as pytest.

GitHub Actions runs your CI tests and provides the results of each test in the pull request, so you can see whether the change in your branch introduces an error:

For more information on the use of status checks, see branch protection.

We did not find any evidence of continuous integration or automated tests for the rdl-standard or rdl-docs repositories.

Recommendations

Use GitHub Actions to set up continuous integration for all commits and pull requests with at least the following checks:

- Check that the RDLS schema is a valid JSON Schema
- Check that the RDLS documentation site builds successfully
- Check that all schema definitions and codelists are included in the documentation site
Semantic versioning

Semantic versioning is a simple set of rules and requirements that dictate how version numbers are assigned in order to convey meaning about what has been modified from one version to the next. In summary, given a MAJOR.MINOR.PATCH version number, increment the:

1. MAJOR version when you make backwards incompatible changes
2. MINOR version when you add functionality in a backwards compatible manner
3. PATCH version when you make backwards compatible bug fixes

MAJOR version zero (0.Y.X) is for initial development and means that anything may change at any time.

In the context of a data standard, semantic versioning ensures that, for each release of the standard, data publishers and users understand whether they need to update their processes and tools. For more information on the practical implementation of semantic versioning, see branches and pull requests and tags.

The governance documentation mentions semantic versioning, however there are no explicit version numbers associated with either the rdl-standard repository or the rdl-docs repository.

Recommendations

Once the project repositories are integrated and normative and non-normative content are clearly separated, agree an initial version number for the RDLS schema and documentation and implement semantic versioning for future changes.

Changelog

A changelog is a file which contains a curated, chronologically ordered list of notable changes for each version of a project. Keeping a changelog makes it easier for users and contributors to see precisely what notable changes have been made between each version of the project.
In the context of a data standard, keeping a changelog ensures that, for each release of the standard, data publishers and users have the information that they need to update their tools and processes.

There is no changelog associated with either the rdl-standard repository or the rdl-docs repository.

Recommendations

Once the project repositories are integrated and normative and non-normative content are clearly separated, document a changelog and keep it up to date for all future changes.

Repositories

A repository contains all of the files for a project and each file's revision history. You can discuss and manage a project within its repository.

The RDLS schema and documentation is split across two repositories:

- rdl-standard is used to coordinate the development of the RDLS data model
- rdl-docs contains the files used to build the RDLS documentation site

Each repository contains a copy of the RDLS schema:

- rdl-standard/blob/main/specs/DDH_compliant/rdl_schema_0_1.json
- rdl-docs/blob/main/docs/schema/rdl_schema_0.1.json

Multiple copies of the schema is a risk to maintaining a single source of truth for the standard.

The best practice for maintaining a SSOT, effective change management and ease of maintenance is to manage the schema and documentation in a single repository. This is the approach used in several other standards:

- Open Contracting Data Standard
- Beneficial Ownership Data Standard
- Open Fibre Data Standard
- 360Giving Standard
## Recommendations

Integrate the content from the rdl-docs repository into the rdl-standard repository.

### Branches and pull requests

Branches allow you to develop features, fix bugs, or safely experiment with new ideas in a contained area of your repository. Each repository has one default branch, and can have multiple other branches. You can merge a branch into another branch using a pull request. In the context of a data standard, branches and pull requests can be used to draft and review changes to the schema and documentation without affecting the live version of the standard.

Best practice is to maintain separate 'live' and development branches for each minor version of the standard, e.g. 1.0 and 1.0-dev. For more information on the meaning of version numbers, see semantic versioning.

To propose a change, a collaborator should create a new branch from 1.0-dev, update the schema or documentation, and open a pull request to merge the change into 1.0-dev.

To release changes to the live version of the standard, a maintainer should open a pull request to merge 1.0-dev into 1.0.

For more information on requiring the use of pull requests, see branch protection.

We reviewed the rdl-standard or rdl-docs repositories and noted that there is no separation of live and development branches. Each repository has a main branch with limited use of branches and pull requests for feature development.

### Recommendations

Until semantic versioning is adopted, maintain separate main and dev branches. Once semantic versioning is adopted, maintain separate live and development branches for each minor version of the standard.

### Pull request template

A pull request template is a file whose contents are automatically added to the description of pull requests. You can use a pull request template to standardise the information that collaborators include when they open a pull request and to ensure that pull requests comply with contributor guidelines. In the context of a data standard, a pull request template can...
help ensure that the changelog is kept up to date and that the documentation is kept in-sync with the schema and codelists.

There is no pull request template for either the `rdl-standard` repository or the `rdl-docs` repository.

**Recommendations**

Document a pull request template based on the examples in Appendix 1: Pull request template examples.

**Branch protection**

Branch protection protects important branches by defining whether collaborators can push directly to the branch and by setting requirements for any pushes to the branch, including merging pull requests. In the context of a data standard, branch protection ensures that collaborators cannot make changes to the schema and documentation without appropriate review and checks.

Best practice is to protect the live and development branches to prevent collaborators from pushing directly to the branch, to require pull request reviews before merging and to require status checks before merging. Pull request reviews allow collaborators to comment on the changes proposed in pull requests, approve the changes, or request further changes before the pull request is merged. For more information on status checks, see continuous integration and automated tests.

The `main` branch in the `rdl-standard` repository is not protected and it appears to be the same for the `rdl-docs` repository.

**Recommendations**

Protect the live and development branches in each repository, require pull request reviews.
before merging and require status checks before merging.

Issue, discussion and pull request labels

Labels are used to categorise issues, pull requests and discussions. In the context of a data standard, labels can be used to categorise whether issues relate to the normative content, non-normative content or the documentation build process.

The rdl-docs repository has the default labels that are added to all new GitHub repositories. However, issues are not categorised against the labels. Issues in the the rdl-standard repository are categorised against the following labels:

- **bug**: Something isn’t working
- **datapackage**: Issues relating to the data package specifications
- **duplicate**: This issue or pull request already exists
- **exposure**: Issues related to Exposure data
- **hazard**: Issues related to Hazard data
- **loss**: Issues related to Loss data
- **metadata**: Issues related to common, core metadata
- **proposal**: New feature or request
- **question**: Further information is requested
- **taxonomy**: Issues related to core taxonomies
- **vulnerability**: Issues related to Vulnerability data
- **wontfix**: This will not be worked on

Recommendations

Once the project repositories are integrated and normative and non-normative content is clearly separated, add labels to distinguish issues related to:

- Normative content
Release tags

Git tags are used to mark a specific point in your repository’s history, typically to mark the release of a new version. In the context of a data standard, tags can be used to refer back to earlier versions of the schema and documentation.

The best practice is to tag each major, minor and patch version release. It is not necessary to tag the release of updates to non-normative content. For more information on the meaning of version numbers, see semantic versioning.

There are no tags in either the rdl-standard repository or the rdl-docs repository.

Recommendations

Once semantic versioning is implemented, tag each major, minor and patch version.

Development

This section documents our assessment of the documentation and methodology in use for the development of the RDLS against best practices, and provides recommendations for improvements.

Build and deploy process

Build and deploy is the process by which source code is compiled into executable software and deployed to a testing or production environment. In the context of a data standard, the process involves building the HTML version of the documentation site from the source files and making it available online.

The build process for the RDLS documentation currently takes place locally on users’ machine with the built documentation deployed from the users’ local machines to GitHub Pages.

A key benefit of automating the build and deploy process is support for multiple versions, including online previews of development branches for testing and review purposes. Development branches can be made available at public URLs, but can be hidden from the version switcher in the live version of the documentation site, e.g.
Other benefits include ensuring that builds are reproducible outside of users' local machines and the option to rollback to an earlier build in the event of unforeseen problems with a release.

Options for automation include using GitHub Actions to deploy to GitHub pages or a web-server, or using Read the Docs to build, deploy and host documentation.

An important consideration when deploying multiple versions is to substitute version and branch names in hardcoded URLs so that users working with online previews of development branches can access the correct version of resources such as schema files.

**Recommendations**

- Use Read the Docs to automate the build, deploy and hosting of the RDLS documentation site.
- Implement version and branch name substitution for development branches.

**Development environment**

A development environment in software and web development is a workspace for developers to make changes without breaking anything in a live environment. In the context of a data standard, a development environment is the collection of tools and processes used by maintainers and collaborators to edit, test and preview the schema and documentation.

There is no development environment specified for the rdl-standard repository. The rdl-docs repository contains a readme file that details the requirements and dependencies for
building, previewing and deploying the documentation. However, there are issues with the instructions as documented:

- How to create and activate a Python virtual environment is not documented
- Python dependencies are not documented in a requirements file and therefore require manual installation
- Python dependencies and are not pinned to specific versions therefore the process for building and deploying the documentation may not be reproducible

More broadly, managing virtual environments and dependencies commonly causes issues for collaborators and maintainers, even for users experienced with technical matters. Similarly, less experienced collaborators may not have a code editor with features appropriate for standard development, e.g. syntax highlighting, auto-indentation, JSON validation, linting and formatting etc.

An alternative to leaving each collaborator to maintain their own development environment on their local machine is to provide a hosted development environment, like GitHub Codespaces. Each codespace runs on a virtual machine hosted by GitHub, addressing issues with managing virtual environments and dependencies. Collaborators can connect using the VS Code Browser in Chrome, Safari or Edge, or using Visual Studio Code, addressing issues with inappropriate code editing tools. You can commit configuration files to your repository to create a repeatable codespace configuration for all collaborators and maintainers of your project.

**Recommendations**

- Document how to create and activate a Python virtual environment
- Document Python dependencies in a requirements file
- Pin Python dependencies to specific versions to ensure documentation builds are reproducible
- Consider configuring a hosted development environment using GitHub Codespaces

**Developer documentation**

Developer documentation for a data standard explains how to develop and maintain the schema and documentation for the standard.

The rdl-docs repository contains a brief readme file that explains:
● how to install the necessary Python dependencies
● the locations of the written content, the theme and the website configuration
● how to build, preview and deploy the documentation

The rdl-standard repository contains high-level contributor documentation aimed at external contributors, that describes how to propose changes.

Whilst the above documentation is certainly useful it is not comprehensive. Providing more comprehensive developer documentation has many benefits:

● Greater consistency in the schema and documentation
● Quicker onboarding for new maintainers and collaborators
● Improved resilience to changes in project staff
● Less time spent on troubleshooting issues

A common approach in other data standards is to document a development handbook, for example:

● Open Contracting Data Standard Development Handbook
● Beneficial Ownership Data Standard Development Handbook
● Open Fibre Data Standard Development Handbook

These handbooks typically cover at least the following topics:

● Style guides describing conventions for writing schema and documentation, GitHub issues, and associated presentations and documents

● The structure and configuration of the standard repository, including the locations of key files and the usage of branches, tags and issues etc.

● Process documentation, covering how to:
  ○ Propose changes
  ○ Review changes
  ○ Perform periodic maintenance tasks
  ○ Build the documentation
  ○ Run tests
  ○ Deploy the documentation
Recommendations

Document a comprehensive development handbook, covering at least the topics listed above.

Other Issues

This section documents our assessment of RDLS with respect to any other issues not already covered in the previous sections. Best practice in standard development and comparisons with robust and mature open data standards are used. Recommendations for improvements are provided.

Translation

Having translations of the standard available in languages other than English can be an important tool to encourage adoption outside of English-speaking countries. This is of mixed importance in RDLS given the wide range of RDM data users. Academics are likely to have English language skills, but local disaster relief agencies working in non-English speaking countries may not. Translation and internationalisation is best done once there is a stable release of the standard.

We have seen no evidence of support for internationalisation or translation of RDLS, but this is typical of a standard in this stage of its development.

Once the standard is further along in its development we would be happy to explore translation and internationalisation options but will not be describing appropriate implementation approaches in this report. Guidance on how to approach translation and internationalisation of a standard can be found in the Open Data Services OD4D handbook.

Tooling

Tooling is currently out of scope but a future report will make tooling recommendations.

Next steps

This section documents the recommended next steps following on from this report.

We have detailed a number of recommendations throughout this report, we will work with the RLDS team and steering committee to agree on the priority of these recommendations.

Immediate next steps to move this work forward are:
<table>
<thead>
<tr>
<th>Task</th>
<th>Timescale</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS to curate a prioritised list of the recommendations within this report with indicative timescales for each task.</td>
<td>End of February 2023</td>
</tr>
<tr>
<td>The steering committee and RDLS team review this report, rejecting or accepting each recommendation.</td>
<td>Mid March 2023</td>
</tr>
<tr>
<td>The steering committee, the RDLS team and Open Data Services agree on the prioritisation of the accepted recommendations.</td>
<td>Mid March 2023</td>
</tr>
<tr>
<td>The RDLS team and Open Data Services agree on a timetabled work plan to implement the prioritised accepted recommendations.</td>
<td>April 2023</td>
</tr>
<tr>
<td>Work begins on implementation of the recommendations.</td>
<td>April - July 2023</td>
</tr>
<tr>
<td>Tooling recommendation report</td>
<td>October 2023</td>
</tr>
</tbody>
</table>
Appendix 1: Pull request template examples

Open Contracting for Infrastructure Data Standard

**Related issues**

<!-- Add links to related issues here. If you want an issue to be automatically closed when the PR is merged, use keywords (https://docs.github.com/en/issues/tracking-your-work-with-issues/linking-a-pull-request-to-an-issue#linking-a-pull-request-to-an-issue-using-a-keyword). -->

**Description**

<!-- If the changes in the PR are not sufficiently explained by the related issues and commit messages, add a description here. -->

**Merge checklist**

<!-- Complete the checklist before requesting a review. -->

- [ ] [Log your changes](https://ocds-standard-development-handbook.readthedocs.io/en/latest/standard/contributing.html#logging-changes)

If there are changes to `project-schema.json` or `project-package-schema.json`:

- [ ] Update the examples:
  - [ ] `docs/examples/example.json`
  - [ ] `docs/examples/blank.json`
- [ ] Run `./manage.py pre-commit` to update `docs/_static/i18n.csv`

If you added a new definition to the schema, update `docs/reference/schema.md`:

- [ ] Add an entry to the components section
- [ ] Update the `:collapse:` parameter of the `jsonschema` directive for any schemas or sub-schemas that reference the new definition

If you added a new codelist:

- [ ] Add an entry to `docs/reference/codelists.md`

Open Fibre Data Standard

**Related issues**

<!-- Add links to related issues here. If you want an issue to be automatically closed when the PR is merged, use keywords (https://docs.github.com/en/issues/tracking-your-work-with-issues/linking-a-pull-request-to-an-issue#linking-a-pull-request-to-an-issue-using-a-keyword) -->

**Description**

<!-- If the changes in the PR are not sufficiently explained by the related issues and commit messages, add a description here -->

**Merge checklist**
<!-- Complete the checklist before requesting a review. -->

- [ ] Run `./manage.py pre-commit` to update derivative schema files, reference documentation and examples

If there are changes to `network-schema.json`, `network-package-schema.json`, `reference/publication_formats/json.md`, `reference/publication_formats/geojson.md` or `guidance/publication.md#how-to-publish-large-networks`, update the relevant manually authored examples:

- [ ] `examples/json/`
  - [ ] `network-package.json`
  - [ ] `api-response.json`
  - [ ] `multiple-networks.json`
  - [ ] `network-embedded.json`
  - [ ] `network-separate-endpoints.json`
  - [ ] `network-separate-files.json`
  - [ ] `nodes-endpoint.json`
  - [ ] `spans-endpoint.json`
- [ ] `examples/geojson/`
  - [ ] `api-response.geojson`
  - [ ] `multiple-networks.geojson`

If you used a validation keyword, type or format that is not [already used in the schema](https://ofds-standard-development-handbook.readthedocs.io/en/latest/standard/schema.html#json-schema-usage):

- [ ] Update the list of validation keywords, types or formats in [JSON Schema]
Add a field that fails validation against the new keyword, type or format to
`network-package-invalid.json`
Check that [OFDS CoVE](https://ofds.cove.opendataservices.coop/) reports an appropriate validation error.

If you added a normative rule that is not encoded in JSON Schema:

- Add a field that does not conform to the rule to `network-package-additional-checks.json`
- Open a [new issue](https://github.com/Open-Telecoms-Data/lib-cove-ofds/issues/new/choose) to add an additional check to Lib Cove OFDS.

If there are changes to `examples/geojson/nodes.geojson` or `examples/geojson/spans.geojson`, check and update the data use examples:

- `examples/leaflet/leaflet.ipynb`
- `examples/qgis/geojson.qgs`
# Appendix 2: Documentation schema vs JSON schema

## Attributes vs Fields

In the following tables an asterisk (*) indicates the attributes identified as required, and attributes with inconsistent names have been paired based on descriptions where possible and best guess otherwise. Any spelling mistakes are present in the source.

### General schema / `contribution`

<table>
<thead>
<tr>
<th>Documentation</th>
<th>rdl_schema_0.1.json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component*</td>
<td>component*</td>
</tr>
<tr>
<td>Source model*</td>
<td>model_source*</td>
</tr>
<tr>
<td>Release date*</td>
<td>model_date*</td>
</tr>
<tr>
<td>Project name</td>
<td>project</td>
</tr>
<tr>
<td>Purpose</td>
<td>purpose</td>
</tr>
<tr>
<td>Notes</td>
<td>notes</td>
</tr>
<tr>
<td>Bibliography</td>
<td>biblio_title</td>
</tr>
<tr>
<td>Bibliography</td>
<td>biblio_url</td>
</tr>
<tr>
<td>Version</td>
<td>version</td>
</tr>
<tr>
<td>Geo coverage*</td>
<td>geo_coverage*</td>
</tr>
<tr>
<td>License code*</td>
<td>license_code*</td>
</tr>
<tr>
<td></td>
<td>title*</td>
</tr>
<tr>
<td></td>
<td>abstract*</td>
</tr>
<tr>
<td></td>
<td>organization*</td>
</tr>
<tr>
<td></td>
<td>publish*</td>
</tr>
<tr>
<td></td>
<td>maintainer</td>
</tr>
<tr>
<td></td>
<td>maintainer_email</td>
</tr>
</tbody>
</table>

### General schema / `resources`

---

Open Data Services Co-operative Limited is a company limited by shares, operated as a Workers’ Co-operative | Registered in England. Company number: 09506232 | Registered address: 1st Floor, Holyoake House, Hanover Street, Manchester, Greater Manchester, M60 0AS | Correspondence to contact@opendataservices.coop
### Documentation

<table>
<thead>
<tr>
<th>Data Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource name</td>
<td>name</td>
</tr>
<tr>
<td>Aggregation type</td>
<td>aggregation_type</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Reference coordinate system</td>
<td>epsg</td>
</tr>
<tr>
<td>Horizontal resolution</td>
<td>h-res</td>
</tr>
<tr>
<td>Format</td>
<td>format</td>
</tr>
<tr>
<td>Download url</td>
<td>url</td>
</tr>
</tbody>
</table>

### Hazard Schema / `hazard`

<table>
<thead>
<tr>
<th>Data Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard type</td>
<td>event_set.hazard_type*</td>
</tr>
<tr>
<td>Analysis type</td>
<td>event_set.analysis_type*</td>
</tr>
<tr>
<td>Calculation method</td>
<td>event.calculation_method*</td>
</tr>
<tr>
<td>Geographic area</td>
<td>event_set.geo_area</td>
</tr>
<tr>
<td>Time start</td>
<td>event_set.time_start</td>
</tr>
<tr>
<td>Time end</td>
<td>event_set.time_end</td>
</tr>
<tr>
<td>Time span</td>
<td>event_set.time_span</td>
</tr>
<tr>
<td>Time year</td>
<td>event_set.time_span</td>
</tr>
<tr>
<td>Frequency type</td>
<td>event_frequency_type*</td>
</tr>
<tr>
<td>Occurrence probability</td>
<td>? event.return_period</td>
</tr>
<tr>
<td>Occurrence time (start)</td>
<td>event.occurrence_time_start</td>
</tr>
<tr>
<td>Occurrence time (end)</td>
<td>event.occurrence_time_end</td>
</tr>
<tr>
<td>Occurrence time (span)</td>
<td>event.occurrence_time_span</td>
</tr>
<tr>
<td>Trigger hazard type</td>
<td>event.trigger_hazard_type</td>
</tr>
<tr>
<td>Trigger process type</td>
<td>event.trigger_process_type</td>
</tr>
<tr>
<td>Hazard process</td>
<td>footprint_set.process_type</td>
</tr>
<tr>
<td>Unit of measure*</td>
<td>footprint_set.imt</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Description</td>
<td>? event.description</td>
</tr>
<tr>
<td>Data uncertainty</td>
<td>footprint_set.data_uncertainty</td>
</tr>
</tbody>
</table>

**Exposure schema / `exposure`**

<table>
<thead>
<tr>
<th>Documentation</th>
<th>rdl_schema_0.1.json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category*</td>
<td>model.category*</td>
</tr>
<tr>
<td>Occupancy*</td>
<td>model.occupancy*</td>
</tr>
<tr>
<td>Occupancy time</td>
<td>model.occupancy_time</td>
</tr>
<tr>
<td>Taxonomy source</td>
<td>model.taxonomy_source</td>
</tr>
<tr>
<td>Taxonomy code</td>
<td>model.taxonomy_code</td>
</tr>
<tr>
<td>Value type*</td>
<td>value.val_type</td>
</tr>
<tr>
<td>Value unit*</td>
<td>value.val_unit</td>
</tr>
<tr>
<td></td>
<td>model.time_year</td>
</tr>
<tr>
<td></td>
<td>model.add_attributes</td>
</tr>
</tbody>
</table>

**Vulnerability schema / `vulnerability`**

<table>
<thead>
<tr>
<th>Documentation</th>
<th>rdl_schema_0.1.json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary hazard*</td>
<td>model.hazard_type_primary</td>
</tr>
<tr>
<td>Secondary hazard</td>
<td>model.hazard_type_secondary</td>
</tr>
<tr>
<td>Primary process*</td>
<td>model.process_type_primary</td>
</tr>
<tr>
<td>Secondary process</td>
<td>model.process_type_secondary</td>
</tr>
<tr>
<td>Frequency*</td>
<td>model.frequency_type</td>
</tr>
<tr>
<td>Intensity unit*</td>
<td>model.imt</td>
</tr>
<tr>
<td>Exposure category*</td>
<td>model.category</td>
</tr>
<tr>
<td>Exposure occupancy*</td>
<td>model.occupancy</td>
</tr>
<tr>
<td></td>
<td>model.val_type</td>
</tr>
<tr>
<td></td>
<td>model.val_unit</td>
</tr>
<tr>
<td>Parameter</td>
<td>Model Attribute</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Taxonomy source</td>
<td>model.taxonomy_source</td>
</tr>
<tr>
<td>Taxonomy code</td>
<td>model.taxonomy_code</td>
</tr>
<tr>
<td>Impact type*</td>
<td>model.impact_type</td>
</tr>
<tr>
<td>Function type*</td>
<td>model.funcation_type</td>
</tr>
<tr>
<td>Approach</td>
<td>model.approach</td>
</tr>
<tr>
<td>[blank]*</td>
<td></td>
</tr>
<tr>
<td>Mathematical model</td>
<td>? model.f_math</td>
</tr>
<tr>
<td>Scale applicability*</td>
<td>model.scale_applicability</td>
</tr>
<tr>
<td>Transferrability*</td>
<td>model.country_tranferability</td>
</tr>
<tr>
<td>Local applicability*</td>
<td>model.local_applicability</td>
</tr>
<tr>
<td>Transferrability notes</td>
<td>model.transferability_notes</td>
</tr>
<tr>
<td>model.f_relationship</td>
<td></td>
</tr>
<tr>
<td>model.calculation_method</td>
<td></td>
</tr>
<tr>
<td>specifics.analysis_details</td>
<td></td>
</tr>
<tr>
<td>par_names</td>
<td>specifics.par_names</td>
</tr>
<tr>
<td>ub_par_value</td>
<td></td>
</tr>
<tr>
<td>ub_par_perc</td>
<td></td>
</tr>
<tr>
<td>med_par_value</td>
<td></td>
</tr>
<tr>
<td>lb_par_value</td>
<td></td>
</tr>
<tr>
<td>lb_par_perc</td>
<td></td>
</tr>
<tr>
<td>damage_scale_code</td>
<td></td>
</tr>
<tr>
<td>dm_state_name</td>
<td></td>
</tr>
<tr>
<td>n_dm_states</td>
<td></td>
</tr>
<tr>
<td>f_disc_im</td>
<td></td>
</tr>
<tr>
<td>f_disc_ep</td>
<td></td>
</tr>
<tr>
<td>lp_code</td>
<td></td>
</tr>
<tr>
<td>lp_loss_value</td>
<td></td>
</tr>
<tr>
<td>edp_code</td>
<td>edp_name</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>im_code</td>
<td>im_name</td>
</tr>
<tr>
<td>im_range</td>
<td>im_units</td>
</tr>
<tr>
<td>im_method</td>
<td>im_sim_type</td>
</tr>
<tr>
<td>impe_referenc</td>
<td>data_countries</td>
</tr>
<tr>
<td>im_data_source</td>
<td>n_events</td>
</tr>
<tr>
<td>n_assets</td>
<td>specifics.n_assets</td>
</tr>
<tr>
<td>nonsampling_err*</td>
<td>type_nonsampling_err</td>
</tr>
<tr>
<td>is_fix_nonsam_err</td>
<td>is_data_aggregated</td>
</tr>
<tr>
<td>is_data_disaggr</td>
<td>n_data_points_aggr</td>
</tr>
<tr>
<td>an_analysis_type</td>
<td>em_analysis_type</td>
</tr>
<tr>
<td>jb_analysis_type</td>
<td>is_fit_good</td>
</tr>
<tr>
<td>fit_ref</td>
<td>val_data_source</td>
</tr>
<tr>
<td>val_study_reference</td>
<td></td>
</tr>
</tbody>
</table>
sample

<table>
<thead>
<tr>
<th>specifics.is_edp_thre</th>
</tr>
</thead>
<tbody>
<tr>
<td>specifics.is_dm_factor</td>
</tr>
<tr>
<td>specifics.is_casualties</td>
</tr>
<tr>
<td>specifics.is_downtime</td>
</tr>
</tbody>
</table>

**Loss schema / 'loss'**

<table>
<thead>
<tr>
<th>Documentation</th>
<th>rdl_schema_0.1.json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazard type*</td>
<td>model.hazard_type</td>
</tr>
<tr>
<td>Hazard process</td>
<td>model.process_type</td>
</tr>
<tr>
<td>Calculation method</td>
<td>model.calculation_method</td>
</tr>
<tr>
<td>Exposure occupancy*</td>
<td>model.occupancy</td>
</tr>
<tr>
<td>Exposure category*</td>
<td>model.category</td>
</tr>
<tr>
<td>Value type*</td>
<td>model.val_type</td>
</tr>
<tr>
<td>Hazard link</td>
<td>model.hazard_link</td>
</tr>
<tr>
<td>Exposure link</td>
<td>model.exposure_link</td>
</tr>
<tr>
<td>Vulnerability link</td>
<td>model.vulnerability_link</td>
</tr>
<tr>
<td>Time start</td>
<td>model.time_start</td>
</tr>
<tr>
<td>Time end</td>
<td>model.time_end</td>
</tr>
<tr>
<td>Time span</td>
<td></td>
</tr>
<tr>
<td>Time year</td>
<td></td>
</tr>
<tr>
<td>Frequency type</td>
<td>model.frequency</td>
</tr>
<tr>
<td>Occurrence probability</td>
<td>? model.return_period</td>
</tr>
<tr>
<td>Impact*</td>
<td>model.impact</td>
</tr>
<tr>
<td>Loss type*</td>
<td>model.loss_type</td>
</tr>
<tr>
<td>Metric*</td>
<td>model.metric</td>
</tr>
<tr>
<td>Unit*</td>
<td>model.val_unit</td>
</tr>
</tbody>
</table>
Type vs enum

Also `common_iso` defining ‘Geo coverage’ and Vulnerability ‘Transferrability’. 

<table>
<thead>
<tr>
<th>Documentation attribute</th>
<th>JSON schema definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Geo coverage</strong> from General</td>
<td>common_iso</td>
</tr>
<tr>
<td><strong>Transferrability</strong> from Vulnerability</td>
<td></td>
</tr>
<tr>
<td><strong>License code</strong> from General</td>
<td>common_license</td>
</tr>
<tr>
<td><strong>Hazard process</strong> from Hazard</td>
<td>common_proces_type</td>
</tr>
<tr>
<td><strong>Primary process</strong> from Vulnerability</td>
<td></td>
</tr>
<tr>
<td><strong>Secondary process</strong> from Vulnerability</td>
<td></td>
</tr>
<tr>
<td><strong>Hazard process</strong> from Loss</td>
<td></td>
</tr>
<tr>
<td><strong>Unit of measure</strong> from Hazard</td>
<td>im_code</td>
</tr>
<tr>
<td><strong>Intensity unit</strong> from Vulnerability</td>
<td></td>
</tr>
<tr>
<td><strong>Hazard type</strong> from Hazard</td>
<td>common_hazard_type</td>
</tr>
<tr>
<td><strong>Trigger hazard type</strong> from Hazard</td>
<td></td>
</tr>
<tr>
<td><strong>Primary hazard</strong> from Vulnerability</td>
<td></td>
</tr>
<tr>
<td><strong>Secondary hazard</strong> from Vulnerability</td>
<td></td>
</tr>
<tr>
<td><strong>Hazard type</strong> from Loss</td>
<td></td>
</tr>
</tbody>
</table>