Contents

1 RO-Crate Metadata Specification 1.0

- Permalink: https://w3id.org/ro/crate/1.0
- Status: Recommendation
- JSON-LD context: https://w3id.org/ro/crate/1.0/context
- This version: https://w3id.org/ro/crate/1.0
- Previous version: https://w3id.org/ro/crate/0.2
- Published: 2019-11-15
- Publisher: researchobject.org community
- Cite as: https://doi.org/10.5281/zenodo.3541888 (this version) https://doi.org/10.5281/zenodo.3406497 (any version)
- Editors: Peter Sefton, Eoghan Ó Carragáin, Stian Soiland-Reyes
- Authors: Peter Sefton, Eoghan Ó Carragáin, Stian Soiland-Reyes, Oscar Corcho, Daniel Garijo, Raul Palma, Frederik Coppens, Carole Goble, José María Fernández, Kyle Chard, Jose Manuel Gomez-Perez, Michael R Crusoe, Ignacio Eguinoa, Nick Juty, Kristi Holmes, Jason A. Clark, Salvador Capella-Gutierrez, Alasdair J. G. Gray, Stuart Owen, Alan R Williams, Giacomo Tartari, Finn Bacall, Thomas Thelen

See https://w3id.org/ro/crate for further details about RO-Crate.

1.1 Introduction & definition of an RO-Crate

> The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in RFC 2119.

This document specifies a method, known as *RO-Crate* (Research Object Crate), of organizing file-based data with associated metadata, using linked data principles, in both human and machine readable formats, with the ability to include additional domain-specific metadata.

The core of RO-Crate is a JSON-LD file, the *RO-Crate Metadata File*, named `ro-crate-metadata.jsonld`. This file contains structured metadata about the dataset as a whole (the *Root Data Entity*) and, optionally, about some or all of its files. This provides a simple way to, for example, assert the authors (e.g. people, organizations) of the RO-Crate or one its files, or to capture more complex provenance for files, such as how they were created using software and equipment.

While providing the formal specification for RO-Crate, this document also aims to be a practical guide for software authors to create tools for generating and consuming research data packages, with explanation by examples.

1.1.1 Terminology

RO-Crate: A directory structure that contains a dataset, which is described in an *RO-Crate Metadata File*.

RO-Crate Root: The top-level directory of the *RO-Crate*, indicated by the presence of the *RO-Crate Metadata File* `ro-crate-metadata.jsonld`

RO-Crate Metadata File: A JSON-LD file stored as `ro-crate-metadata.jsonld` in the *RO-Crate Root*. The metadata file describes the *RO-Crate* with structured data in form of *RO-Crate JSON-LD*.

RO-Crate Website: Human-readable HTML pages which describe the RO-Crate (i.e. the *Root Data Entity*, its *Data Entities* and *Context Entities*), with a home-page at `ro-crate-preview.html` (any additional files reside in `ro-crate-preview_files/`)

Data Entity: A JSON-LD representation, in the *RO-Crate Metadata File*, of a directory, file or other resource contained or described by the RO-Crate.

Root Data Entity: A *Data Entity* of type Dataset, representing the RO-Crate as a whole.

RO-Crate Metadata File Descriptor: A *Contextual Entity* of type CreativeWork, which describes the *RO-Crate Metadata File* and links it to the *Root Data Entity*.

JSON-LD: A JSON-based file format for storing *Linked Data*. This document assumes JSON-LD 1.0. JSON-LD use a *context* to map from JSON keys to *URIs*.

JSON: The *JavaScript Object Notation (JSON) Data Interchange Format* as defined by RFC 7159; a structured text file format that can be programmatically consumed and generated in a wide range of programming languages.

Contextual Entity: A JSON-LD representation of an entity associated with a *Data Entity*, needed to adequately describe that *Data Entity*. For example, a Person, Organization (including research projects), item of equipment (IndividualProduct), license or any other *thing* or *event* that forms part of the metadata for a *Data Entity* or supporting information.

Linked Data: A data structure where properties, types and resources are identified with *URIs*, which if retrieved over the Web, further describe or provide the identified property/type/resource.

URI: A *Uniform Resource Identifier* as defined in RFC 3986, for example `http://example.com/path/file.html` - commonly known as *URL*. In this

document the term *URI* includes *IRI*, which also permit international Unicode characters.

URI Path: The relative *path* element of an *URI* as defined in RFC3986 section 3.3, e.g. `path/file.html`

RO-Crate JSON-LD Context: A JSON-LD context that provides Linked Data mapping for RO-Crate metadata to vocabularies like schema.org.

RO-Crate JSON-LD: JSON-LD structure using the *RO-Crate JSON-LD Context* and containing RO-Crate metadata, written as if flattened and then compacted according to the rules in JSON-LD 1.0. The *RO-Crate JSON-LD* for an *RO-Crate* is stored in the *RO-Crate Metadata File*.

1.1.2 Linked Data conventions

Throughout this specification, RDF terms are referred to using the keys defined in the *RO-Crate JSON-LD Context*.

Following schema.org practice, `property` names start with lowercase letters and `Class` names start with uppercase letters.

In `ro-crate-metadata.jsonld` the RDF terms use their RO-Crate JSON-LD names as defined in the *RO-Crate JSON-LD Context*, which is available at https://w3id.org/ro/crate/1.0/context

1.2 RO-Crate Structure

The structure an *RO-Crate* MUST follow is:

```
<RO-Crate root directory>/
|   ro-crate-metadata.jsonld  # RO-Crate Metadata File MUST be present
|   ro-crate-preview.html     # RO-Crate Website homepage MAY be present
|   ro-crate-preview_files/   # MAY be present
|    | [other RO-Crate Website files]
|   [payload files and directories]  # 1 or more SHOULD be present
```

The name of the *RO-Crate root* directory is not defined, but a root directory is identifiable by the presence of `ro-crate-metadata.jsonld`. For instance, if an *RO-Crate* is archived in a ZIP-file, the ZIP root directory correspond to *RO-Crate root* directory if it contains `ro-crate-metadata.jsonld`.

1.2.1 RO-Crate Metadata File (`ro-crate-metadata.jsonld`)

- The *RO-Crate Metadata File* MUST be named `ro-crate-metadata.jsonld` and appear in the *RO-Crate Root*

- The *RO-Crate Metadata File* MUST contain *RO-Crate JSON-LD*; a valid JSON-LD 1.0 document in flattened and compacted form
- The *RO-Crate JSON-LD* SHOULD use the *RO-Crate JSON-LD Context* https://w3id.org/ro/crate/1.0/context by reference.

JSON-LD is a structured form of JSON that can represent a *Linked Data* graph.

A valid *RO-Crate JSON-LD* graph MUST describe:

1. The *RO-Crate Metadata File Descriptor*
2. The *Root Data Entity*
3. Zero or more *Data Entities*
4. Zero or more *Contextual Entities*

It is RECOMMENDED that any referenced *contextual entities* are also described in the *RO-Crate Metadata File* with the same identifier. Similarly it is RECOMMENDED that any *contextual entity* in the *RO-Crate Metadata file* is linked to from at least one of the other entities using the same identifier.

The appendix RO-Crate JSON-LD details the general structure of the JSON-LD that is expected in the *RO-Crate Metadata File*. In short, the rest of this specification describe the different types of entities that can be added as `{}` objects to the *RO-Crate JSON-LD* `@graph` below:

```
{ "@context": "https://w3id.org/ro/crate/1.0/context",
  "@graph": [

  ]
}
```

1.2.2 RO-Crate Website (`ro-crate-preview.html` and `ro-crate-preview_files/`)

In addition to the machine-oriented *RO-Crate Metadata File*, the RO-Crate MAY include a human-readable HTML rendering of the same information, known as the *RO-Crate Website*.

If present in the root directory, `ro-crate-preview.html` MUST:

- Be a valid HTML 5 document
- Contain at least a human readable summary of metadata relating to the *Root Data Entity*
- Contain a copy of the *RO-Crate JSON-LD* in a `script` element of the `head` element of the HTML, for example:

```
<script type="application/ld+json">
{
    "@context": "https://w3id.org/ro/crate/1.0/context",
    "@graph": [ ...]
```

```
}
</script>
```

`ro-crate-preview.html` SHOULD:

- Contain at least the same information as the *RO-Crate JSON-LD*, with the exception that files which have no description, author or similar metadata MAY not be listed in the website.
- Display at least the metadata relating to the *Root Data Enity* as static HTML without the need for scripting. It MAY contain extra features enabled by JavaScript.
- When a *Data Entity* or *Contextual Entity* is referenced by its ID:
- If it has a name property, provide a link to its HTML version.
- If it does not have a name (e.g. a GeoCoordinates location), show it embedded in the HTML for the entity.
- For keys that resolve in the `RO-Crate JSON-LD Context` to a URI, indicate this (the simplest way is to link the key to its definition.
- For external URI values, provide a link.
- If there is sufficient metadata, contain a prominent *"Cite-as"* text with a natural language data citation (see for example the FORCE11 Data Citation Principles).
- If there are additional resources necessary to render the preview (e.g. CSS, JSON, HTML), link to them in a subdirectory `ro-crate-preview-files/`

1.2.3 Payload files and directories

These are the actual files and directories that make up the dataset being described.

The base RO-Crate specification makes no assumptions about the presence of any specific files or folders beyond the reserved RO-Crate files described above. Payload files may appear directly in the *RO-Crate Root* alongside the *RO-Crate Metadata File*, and/or appear in sub-directories of the *RO-Crate Root*. Each file and directory MAY be represented as Data Entities in the *RO-Crate Metadata File*.

1.2.4 RO-Crates SHOULD be self-describing and self-contained

A minimal RO-Crate is a directory containing a single *RO-Crate Metadata File*.

At the basic level, an RO-Crate is a collection of files represented as a schema.org Dataset, that together form a meaningful unit for the purposes of communication, citation, distribution, preservation, etc. The *RO-Crate Metadata File* describes the RO-Crate, and MUST be stored in the *RO-Crate Root*. Self-containment is a core principle of RO-Crate, i.e. that all *Dataset* files and relevant metadata SHOULD, as far as possible, be contained by the RO-Crate, rather than referring to external resources. However the RO-Crate MAY also reference external

resources which are stored or accessed separately, via URIs, e.g. because these cannot be included for practical or legal reasons.

It is important to note that the *RO-Crate Metadata File* is not an exhaustive manifest or inventory, that is, it does not necessarily list or describe all files in the package. Rather it is focused on providing sufficient amount of metadata to understand and use the content, and is designed to be compatible with existing and future approaches that *do* have full inventories / manifest and integrity checks, e.g. by using checksums, such as BagIt and Oxford Common File Layout OCFL Objects.

The intention is that RO-Crates can work well with a variety of archive file formats, e.g. tar, zip, etc., and approaches to capturing file manifests and file fixity, such as BagIt, OCFL and git.

1.3 RO-Crate Metadata

RO-Crate aims to capture and describe the Research Object using structured *metadata*.

The *RO-Crate Metadata File Descriptor* contains the metadata that describes the RO-Crate and its content, in particular:

- Root Data Entity - the `Dataset` itself, a gathering of data
- Data Entities - the *data* payload, in the form of files and folders
- Contextual Entities - related things in the world (e.g. people, organizations, places), providing provenance for the data entities and the RO-Crate.

This machine-readable metadata can also be represented for human consumption in the *RO-Crate Website*, linking to data and Web resources.

1.3.1 RO-Crate uses Linked Data principles

RO-Crate makes use of Linked Data principles for its description. In particular:

1. (Meta)data should be made available as **Open Data** on the web.
2. (Meta)data should be **machine-readable** in a structured format.
3. (Meta)data should *not* require proprietary software packages.
4. (Meta)data should use open standards from W3C, such as RDF and SPARQL.
5. (Meta)data should **link** to other people's data to provide context, using *URIs* as global identifiers

RO-Crate realize these principles using a particular set of technologies and best practices:

1. The *RO-Crate Metadata File* and *RO-Crate Website* can be directly published on the web together with the RO-Crate payload. In addition, a data

package (e.g. BagIt Zip archive) that contain the RO-Crate can also be published on the web.
2. The *RO-Crate Metadata File* is based on the structured data format JSON.
3. Multiple open source tools/libraries are available for JSON and for JSON-LD.
4. The *RO-Crate Website* is HTML 5, and the *RO-Crate Metadata File* is JSON-LD, one of the W3C RDF 1.1 formats.
5. The *RO-Crate Metadata File* reuse common vocabularies like schema.org, and this specification recommend identifiers it should link to.

1.3.2 Base metadata standard for RO-Crate: Schema.org

schema.org is the base metadata standard for RO-Crate. Schema.org was chosen because it is widely used on the World Wide Web and supported by search engines, on the assumption that discovery is likely to be maximized if search engines index the content. NOTE: As far as we know there is no alternative, well-maintained linked-data schema for research data with the coverage needed for this project - i.e. a single standard for expressing all the examples presented in this specification.

RO-Crate relies heavily on schema.org using a constrained subset of JSON-LD, and this document gives opinionated recommendations on how to represent the metadata using existing linked data best practices.

1.3.2.1 Differences from schema.org Generally, the standard keys from schema.org should be used. However, RO-Crate uses variant names for some elements, specifically:

- `File` is mapped to http://schema.org/MediaObject which was chosen as a compromise as it has many of the properties that are needed to describe a generic file. Future versions of schema.org or a research data extension may re-define `File`.
- `Journal` is mapped to http://schema.org/Periodical.

Note that JSON-LD examples given on http://schema.org/ website may not be in *flattened* form; any nested entities in *RO-Crate JSON-LD* SHOULD be described as separate contextual entities in the flat `@graph` list.

To simplify processing and avoid confusion with string values, the *RO-Crate JSON-LD Context* requires URIs and entity references to be given in the form `"author": {"@id": "http://example.com/alice"}`, even where schema.org for some properties otherwise permit shorter forms like `"author": "http://example.com/alice"`.

See the appendix RO-Crate JSON-LD for details.

1.3.3 Additional metadata standards

The following terms from the from the Research Object ontologies are used:

- `Workflow` for a workflow definition, mapped to http://purl.org/wf4ever/wfdesc#Workflow
- `Script` for a script, mapped to http://purl.org/wf4ever/wf4ever#Script
- `WorkflowSketch` for an image depicting a workflow, mapped to http://purl.org/wf4ever/roterms#Sketch.

RO-Crate also uses the *Portland Common Data Model* (PCDM)) and imports these terms:

- `RepositoryObject` mapped to https://pcdm.org/2016/04/18/models#Object
- `RepositoryCollection` mapped to https://pcdm.org/2016/04/18/models#Collection
- `RepositoryFile` mapped to https://pcdm.org/2016/04/18/models#Collection
- `hasMember` mapped to https://pcdm.org/2016/04/18/models#hasMember
- `hasFile` mapped to https://pcdm.org/2016/04/18/models#hasFile

The keys `RepositoryObject` and `RepositoryCollection` were chosen to avoid collision between the terms Collection and Object with other vocabularies.

From Dublin Core Terms RO-Crate use:

- `conformsTo` mapped to http://purl.org/dc/terms/conformsTo

1.3.4 Summary of Coverage

RO-Crate is simply a way to make metadata assertions about a set of files and folders that make up a *Dataset*. These assertions can be made at three levels:

- Assertions at the RO-Crate level: for an RO-Crate to be useful, some metadata should be provided about the dataset as a whole (see minimum requirements for different use-cases below). In the *RO-Crate Metadata File*, we distinguish the *Root Data Entity* which represents the RO-Crate as a whole, from other *Data Entities* (files and folders contained in the RO-Crate) and *Contextual Entities*, e.g. a person, organisation, place related to an RO-Crate *Data Entity*
- Assertions about files and folders contained in the RO-Crate: in addition to providing metadata about the RO-Crate as a whole, RO-Crate allows metadata assertions to be made about any other *Data Entity*

This document has guidelines for ways to represent common requirements for describing data in a research context, e.g.:

- Contact information for a data set.

- Descriptive information for a dataset and the files within it and their contexts such as an abstract, spatial and temporal coverage.
- Associated Publications.
- Funding relationships.
- Provenance information of various kinds; who (people and organizations) and what (instruments and computer programs) created or contributed to the data set and individual files within it.
- Workflows that operate on the data using standard workflow descriptions including 'single step workflows'; executable files or environments such as singularity containers or Jupyter notebooks.

However, as RO-Crate uses *Linked Data* principles, adopters of RO-Crate are free to supplement RO-Crate using schema.org metadata and/or assertions using other *Linked Data* vocabularies.

1.3.5 Future coverage

A future version of this specification will allow for variable-level assertions: In some cases, e.g. for tabular data, additional metadata may be provided about the structure and variables within a given file see the Use Case Describe a tabular data file directly in RO-Crate metadata for work-in-progress.

1.3.6 Recommended Identifiers

RO-Crate JSON-LD SHOULD use the following IDs where possible:

- For a *Root Data Entity*, an `identifier` which is RECOMMENDED to be a https://doi.org/ URI.
- For a Person participating in the research process: ORCID identifiers, e.g. https://orcid.org/0000-0002-1825-0097
- For Organizations including funders, Research Organization Registry URIs, e.g. https://ror.org/0384j8v12
- For items of type Place, a geonames URL, e.g. http://sws.geonames.org/8152662/
- For file formats, a Pronom URL, for example https://www.nationalarchives.gov.uk/PRONOM/fmt/831.

In the absence of the above, RO-Crates SHOULD contain stable persistent URIs to identify all entities wherever possible.

1.3.7 Core Metadata for the *Root Data Entity*

The *Root Data Entity* is a Dataset that represent the RO-Crate as a whole; a *Research Object* that includes the *Data Entities* and the related *Contextual Entities*.

The *RO-Crate JSON-LD* MUST contain a *RO-Crate Metadata File Descriptor* with the `@id` value `ro-crate-metadata.jsonld` and `@type` CreativeWork. This descriptor MUST have an about property referencing the *Root Data Entity*, which SHOULD have an `@id` of `./`.

```
{ "@context": "https://w3id.org/ro/crate/1.0/context",
  "@graph": [
    {
        "@type": "CreativeWork",
        "@id": "ro-crate-metadata.jsonld",
        "conformsTo": {"@id": "https://w3id.org/ro/crate/1.0"},
        "about": {"@id": "./"}
    },

    {
      "@id": "./",
      "@type": "Dataset",
      ...
    }
  ]
}
```

The conformsTo of the *RO-Crate Metadata File Descriptor* SHOULD be a versioned permalink URI of the RO-Crate specification that the *RO-Crate JSON-LD* conforms to. The URI SHOULD start with `https://w3id.org/ro/crate/`.

Consumers processing the RO-Crate as an JSON-LD graph can thus reliably find the the *Root Data Entity* by following this algorithm:

1. For each entity in `@graph` array
2. ..if the `conformsTo` property is a URI that starts with `https://w3id.org/ro/crate/`
3. ....from this entity's `about` object keep the `@id` URI as variable *root*
4. For each entity in `@graph` array
5. .. if the entity has an `@id` URI that matches *root* return it

To ensure a base-line interoperability between RO-Crates, and for an RO-Crate to be considered a *Valid RO-Crate*, a minimum set of metadata is required for the *Root Data Entity*. As stated above the *RO-Crate Metadata File* is not an exhaustive manifest or inventory, that is, it does not necessarily list or describe all files in the package. For this reason, there are no minimum metadata requirements in terms of describing *Data Entities* (files and folders) other than the *Root Data Entity*. Extensions of RO-Crate dealing with specific types of dataset may put further constraints or requirements of metadata beyond the Root Data Entity (see Extending RO-Crate below).

The *RO-Crate Metadata File Descriptor* MAY contain information such as licensing for the *RO-Crate Metadata File* so metadata can be licensed separately from Data.

The table below outlines the properties that the *Root Data Entity* MUST have to be minimally valid and additionally highlights properties required to meet other common use-cases, including the minimum metadata necessary to mint a DataCite DOI:

1.3.7.1 Direct properties of the Root Data Entity

The *Root Data Entity* MUST have the following properties:

- `@type`: MUST be Dataset
- `@id`: MUST be a a string of './'
- `name`: SHOULD identify the dataset to humans well enough to disambiguate it from other RO-Crates
- `description`: SHOULD further elaborate on the name to provide a summary of the context in which the dataset is important.
- `datePublished`: MUST be a string in ISO 8601 date format and SHOULD be specified to at least the precision of a day, MAY be a timestamp down to the millisecond.
- `license`: SHOULD link to a *Contextual Entity* in the *RO-Crate Metadata File* with a name and description. MAY have a URI (eg for Creative Commons or Open Source licenses). MAY, if necessary be a textual description of how the RO-Crate may be used.

NOTE: These requirements are stricter than those published for Google Dataset Search which requires a `Dataset` to have a `name` and `description`,

NOTE: The properties above are not sufficient to generate a DataCite citation. Advice on integrating with DataCite will be provided in a future version of this specification, or as an implementation guide.

The following *RO-Crate Metadata File* represents a minimal description of an *RO-Crate*.


```
{ "@context": "https://w3id.org/ro/crate/1.0/context",
  "@graph": [

 {
    "@type": "CreativeWork",
    "@id": "ro-crate-metadata.jsonld",
    "conformsTo": {"@id": "https://w3id.org/ro/crate/1.0"},
    "about": {"@id": "./"}
 },
 {
    "@id": "./",
    "identifier": "https://doi.org/10.4225/59/59672c09f4a4b",
    "@type": "Dataset",
    "datePublished": "2017",
    "name": "Data files associated with the manuscript:Effects of facilitated family case co
```

```
    "description": "Palliative care planning for nursing home residents with advanced dement
    "license": {"@id": "https://creativecommons.org/licenses/by-nc-sa/3.0/au/"}
 },
{
  "@id": "https://creativecommons.org/licenses/by-nc-sa/3.0/au/",
  "@type": "CreativeWork",
  "description": "This work is licensed under the Creative Commons Attribution-NonCommercial
  "identifier": "https://creativecommons.org/licenses/by-nc-sa/3.0/au/",
  "name": "Attribution-NonCommercial-ShareAlike 3.0 Australia (CC BY-NC-SA 3.0 AU)"
  }
 ]
}
```

1.3.8 Examples of referencing *Data Entities* (files and folders) from the *Root Data Entity*

Where files and folders are represented as *Data Entities* in the RO-Crate JSON-LD, these MUST be linked to, either directly or indirectly, from the Root Data Entity using the hasPart property. Directory hierarchies MAY be represented with nested Dataset *Data Entities*, or the Root Dataset MAY refer to files anywhere in the hierarchy using hasPart.

Data Entities representing files MUST be of `@type: File`, which is an RO-Crate alias for http://schema.org/MediaObject

Data Entities representing directories MUST be of `@type: Dataset`.

Note: as indicated above, there is no requirement to represent every file and folder in an RO-Crate as Data Entities in the RO-Crate JSON-LD.

1.3.8.1 Example linking to a file and folders

```
<base directory>/
  |   ro-crate-metadata.jsonld
  |   cp7glop.ai
  |   lots_of_little_files/
  |    | file1
  |    | file2
  |    | ...
  |    | file54
```

An example *RO-Crate JSON-LD* for the above would be as follows:

```
{ "@context": "https://w3id.org/ro/crate/1.0/context",
  "@graph": [
    {
        "@type": "CreativeWork",
```

```
        "@id": "ro-crate-metadata.jsonld",
        "conformsTo": {"@id": "https://w3id.org/ro/crate/1.0"},
        "about": {"@id": "./"}
  },
  {
    "@id": "./",
    "@type": [
      "Dataset"
    ],
    "hasPart": [
      {
        "@id": "cp7glop.ai"
      },
      {
        "@id": "lots_of_little_files/"
      },
      ],
   },
  {
    "@id": "cp7glop.ai",
    "@type": "File",
    "contentSize": "383766",
    "description": "Illustrator file for Glop Pot",
    "encodingFormat": "application/pdf"
  },
  {
      "@id": "lots_of_little_files/",
      "@type": "Dataset",
      "description": "This directory contains many small files, that we're not going to desc
      "name": "Too many files",

  }
]
```

1.3.8.2 Adding detailed descriptions of encodings

The above example provides a media type for the file `cp7glop.ai` - which is useful as it may not be apparent that the file readable as a PDF file from the extension. To add more detail, encodings SHOULD be linked using a PRONOM identifier to a *Contextual Entity* of `@type` Website.

```
 {
    "@id": "cp7glop.ai",
    "@type": "File",
    "contentSize": "383766",
    "description": "Illustrator file for Glop Pot",
```

```
    "encodingFormat": ["application/pdf", {"@id": "https://www.nationalarchives.gov.uk/PRONO
  },
  {
    "@id": "https://www.nationalarchives.gov.uk/PRONOM/fmt/19",
    "name": "Acrobat PDF 1.5 - Portable Document Format",
    "@type": "Website"
  }
```

If there is no PRONOM identifier, then a contextual entity with a URL as an `@id` MAY be used:

For example:

```
 {
    "@id": "1st-tool.cwl",
    "@type": "File",
    "contentSize": "120",
    "description": "An example Common Workflow Language File",
    "encodingFormat": ["text/plain", {"@id": "https://www.commonwl.org/v1.0/Workflow.html"}]
  },
    {
    "@id": "https://www.commonwl.org/v1.0/Workflow.html",
    "name": "Common Workflow Language (CWL) Workflow Description, v1.0.2",
    "@type": "Website"
  }
```

If there is no web-accessible description for a file format it SHOULD be described locally in the dataset, for example in a file:

```
 {
    "@id": "some-file.some_extension",
    "@type": "File",
    "contentSize": "120",
    "description": "A file in a non-standard format",
    "encodingFormat": ["text/plain", {"@id": "https://www.commonwl.org/v1.0/Workflow.html"}]
  },
  {
    "@id": "some_extension.md",
    "encodingFormat": "text/markdown",
    "name": "Description of some_extension file format",
    "@type": ["File", "CreativeWork"]
  }
```

1.3.9 Core Metadata for *Data Entities*

The table below outlines the properties that Data Entities, when present, MUST have to be minimally valid .

1.3.9.1 File Data Entity A `File` *Data Entity* MUST have the following properties:

- `@type`: MUST be `File`, or an array where `File` is one of the values.
- `@id` MUST be a *URI Path* relative to the *RO Crate root*

1.3.9.2 Directory File Entity A `Dataset` (directory) *Data Entity* MUST have the following properties:

- `@type` MUST be `Dataset` or an array wher `Dataset` is one of the values.
- `@id` MUST be a *URI Path* relative to the _RO Crate root; SHOULD end with `/`

1.4 Representing *Contextual Entities*

The *RO-Crate JSON-LD* @graph SHOULD contain additional information about *Contextual Entities* for the use of both humans (in `ro-crate-preview.html`) and machines (in `ro-crate-metadata.jsonld`). This also helps to maximise the extent to which an *RO-Crate* is self-contained and self-describing, in that it reduces the need for the consumer of an RO-Crate to refer to external information which may change or become unavailable over time.

1.4.1 People

A core principle of Linked data is to use URIs to identify things such as people. The following is the minimum recommended way of representing a author in a RO-Crate. This property MAY be applied in the context of a directory (Dataset) or to a File.

```
{
    "@type": "Dataset",
    "@id": "./",
    "author": {"@id": "https://orcid.org/0000-0002-8367-6908"}
}
{
    "@id": "https://orcid.org/0000-0002-8367-6908",
    "@type": "Person",
    "affiliation": "University of Technology Sydney",
```

```
    "name": "J. Xuan"
}
```

This uses an ORCID to unambiguously identify an author, with a *Contextual Entity* of type Person.

Note the string-value of the organizational affiliation. This SHOULD be improved by also providing a *Contextual Entity* for the organization (see example below).

1.4.2 Organizations as values

An Organization SHOULD be the value for the publisher property of a Dataset or ScholarlyArticle or affiliation property of a Person.

```
{
  "@type": "Dataset",
  "@id": "./",
  "publisher": {"@id": "https://ror.org/03f0f6041"}
}

{
  "@id": "https://ror.org/03f0f6041",
  "@type": "Organization",
  "name": "University of Technology Sydney",
  "url": "https://ror.org/03f0f6041"
}
```

An Organization SHOULD also be used for a Person's affiliation property.

```
{
  "@type": "Dataset",
  "@id": "./",
  "publisher": {"@id": "https://ror.org/03f0f6041"},
  "author": {"@id": "https://orcid.org/0000-0002-3545-944X"}
},
{
  "@id": "https://ror.org/03f0f6041",
  "@type": "Organization",
  "name": "University of Technology Sydney"
},
{
  "@id": "https://orcid.org/0000-0002-3545-944X",
  "@type": "Person",
  "affiliation": {"@id": "https://ror.org/03f0f6041"},
  "email": "peter.sefton@uts.edu.au",
  "name": "Peter Sefton"
}
```

1.4.3 More detail on ContactPoint

A RO-Crate SHOULD have contact information, using a contextual entity of type ContactPoint. Note that in schema.org Dataset does not currently have the corresponding contactPoint property, so the contact point would need to be given through a Person or Organization contextual entity which are related to the Dataset via a author or publisher property.

```
{
      "@id": "./",
      "@type": "Dataset",
      "author": {"@id": "https://orcid.org/0000-0001-6121-5409"}
},
{
      "@id": "https://orcid.org/0000-0001-6121-5409",
      "@type": "Person",
      "contactPoint": {
        "@id": "mailto:tim.luckett@uts.edu.au"
      },
      "familyName": "Luckett",
      "givenName": "Tim",
      "identifier": "https://orcid.org/0000-0001-6121-5409",
      "name": "Tim Luckett"
},
{
      "@id": "mailto:tim.luckett@uts.edu.au",
      "@type": "ContactPoint",
      "contactType": "customer service",
      "email": "tim.luckett@uts.edu.au",
      "identifier": "tim.luckett@uts.edu.au",
      "url": "https://orcid.org/0000-0001-6121-5409"
}
```

1.4.4 Publications via citation property

To associate a publication with a dataset the *RO-Crate JSON-LD* MUST include a URL (for example a DOI URL) as the '@id of a publication using the citation property.

For example:

```
"citation": {"@id": "https://doi.org/10.1109/TCYB.2014.2386282"}
```

The publication SHOULD be described in the *RO-Crate JSON-LD*.

```
{
  "@id": "https://doi.org/10.1109/TCYB.2014.2386282",
```

```
  "@type": "ScholarlyArticle",
  "author": [
    {
      "@id": "https://orcid.org/0000-0002-8367-6908"
    },
    {
      "@id": "https://orcid.org/0000-0003-0690-4732"
    },
    {
      "@id": "https://orcid.org/0000-0003-3960-0583"
    },
    {
      "@id": "https://orcid.org/0000-0002-6953-3986"
    }
  ],
  "identifier": "https://doi.org/10.1109/TCYB.2014.2386282",
  "issn": "2168-2267",
  "name": "Topic Model for Graph Mining",
  "journal": "IEEE Transactions on Cybernetics",
  "datePublished": "2015"
}
```

1.4.5 Publisher

The *Root Data Entity* SHOULD have a publisher property. This SHOULD be an Organization though it MAY be a Person.

```
{
  "@id": "https://doi.org/10.5281/zenodo.1009240",
  "@type": "Dataset",
  "name": "Sample dataset for RO-Crate v0.2",
  "publisher": {
    "@id": "https://ror.org/03f0f6041"
  },
  "temporalCoverage": "2017"
},

{
  "@id": "https://ror.org/03f0f6041",
  "@type": "Organization",
  "identifier": "https://ror.org/03f0f6041",
  "name": "University of Technology Sydney"
},
```

1.4.6 Funding and grants

To associate a research project with a Dataset, the *RO-Crate JSON-LD* SHOULD contain an entity for the project using type Organization, referenced by a funder property. The project `Organization` SHOULD in turn reference any external funder, either by using its URL as an `@id` or via a *Contextual Entity* describing the funder.

NOTE: To make it very clear where funding is coming from, the *Root Data Entity* SHOULD also reference funders directly, as well as via a chain of references.

```
{
  "@id": "https://doi.org/10.5281/zenodo.1009240",
  "@type": "Dataset",
  "funder": {
    "@id": "https://ror.org/038sjwq14"
  },
},
{
  "@id": "https://eresearch.uts.edu.au/projects/provisioner",
  "@type": "Organization",
  "description": "The University of Technology Sydney Provisioner project is ...",
  "funder": [
    {
      "@id": "https://ror.org/03f0f6041"
    },
    {
      "@id": "https://ands.org.au"
    }
  ],
  "identifier": "https://eresearch.uts.edu.au/projects/provisioner",
  "name": "Provisioner"
},
{
  "@id": "https://ror.org/03f0f6041",
  "@type": "Organisation",
  "identifier": "https://ror.org/03f0f6041",
  "name": "University of Technology Sydney"
},
{
  "@id": "https://ands.org.au",
  "@type": "Organization",
  "description": "The core purpose of the Australian National Data Service (ANDS) is ...",
  "identifier": "https://ands.org.au",
  "name": "Australian National Data Service"
},
```

1.4.7 Licensing, Access control and copyright

If a *Data Entity* has a license that is different from the license on the *Root Data Entity*, the entity SHOULD have a license property referencing a *Contextual Entity* with a type CreativeWork to describe the license. The `@id` of the license SHOULD be its URL (e.g. a Creative Commons License URL) and, when possible, a summary of the license included using the description property.

The below *Data Entity* has a copyrightHolder which is different from its author. There is a reference to an Organization describing the copyright holder and, to give credit, a sameAs relation to a web page. The license property here refers to https://creativecommons.org/licenses/by/4.0/ which is expanded in a separate contextual entity.

```
{
  "@id": "SciDataCon Presentations/AAA_Pilot_Project_Abstract.html",
  "@type": "File",
  "contentSize": "17085",
  "copyrightHolder": {
    "@id": "https://www.idrc.ca/"
  },
  "author": {
    "@id": "https://orcid.org/0000-0002-0068-716X"
  },
  "description": "Abstract for the Pilot Project initial findings",
  "encodingFormat": "text/html",
  "license": {
    "@id": "https://creativecommons.org/licenses/by/4.0/"
  },
  "sameAs": "https://www.scidatacon.org/2016/sessions/56/paper/265/"
},

{
  "@id": "https://creativecommons.org/licenses/by/4.0/",
  "@type": "CreativeWork",
  "name": "CC BY 4.0",
  "description": "Creative Commons Attribution 4.0 International License"
},

{
  "@id": "https://orcid.org/0000-0002-0068-716X",
  "@type": "Person",
  "identifier": "https://orcid.org/0000-0002-0068-716X",
  "name": "Cameron Neylon"
},
```

```
{
  "@id": "https://www.idrc.ca/",
  "@type": "Organization",
  "description": "Canadian Frown Corporation and funder of development research",
  "identifier": "IDRC",
  "name": "International Development Research Center"
}
```

1.4.7.1 Metadata license In some cases the license of the RO-Crate metadata the (JSON-LD statements in the *RO-Crate Metadata File Descriptor*) is different from the license on the *Root Date Entity* and its content (*data entities* indicated by hasPart).

For instance, a common pattern for repositories is to license metadata as CC0 Public Domain Dedication, while data is licensed as CC-BY or similar. This pattern allow metadata to be combined freely (e.g. the DataCite knowledge graph), while redistribution of data files would require explicit attribution and statement of their license.

To express the metadata license is different from the *Root Data Entity*, expand the *RO-Crate Metadata File Descriptor* to include `license`:

```
{
  "@type": "CreativeWork",
  "@id": "ro-crate-metadata.jsonld",
  "identifier": "ro-crate-metadata.jsonld",
  "about": {"@id": "./"},
  "license": {
    "@id": "https://creativecommons.org/publicdomain/zero/1.0/"
  }
},

{
  "@id": "./",
  "@type": "Dataset",
  "license": {
    "@id": "https://creativecommons.org/licenses/by/4.0/"
  }
}
```

If no explicit `license` is expressed on the *RO-Crate Metadata File Descriptor*, the `license` expressed on the *Root Data Entity* apply also on the RO-Crate metadata.

1.4.8 Provenance: Equipment used to create files

To specify which equipment was used to create or update a *Data Entity*, the *RO-Crate* JSON-LD SHOULD have a *Context Entity* for each item of equipment which SHOULD be of `@type` IndividualProduct. The entity SHOULD have a serial number, manufacturer that identifies it as completely as possible. In this case the equipment is a bespoke machine. The equipment SHOULD be described on a web page, and the address of the description SHOULD be used as its `@id`.

```
{
  "@id": "https://confluence.csiro.au/display/ASL/Hovermap",
  "@type": "IndividualProduct",
  "description": "The CSIRO bentwing is an unmanned aerial vehicle (UAV, commonly known as a
  "identifier": "https://confluence.csiro.au/display/ASL/Hovermap",
  "name": "Bentwing"
}
```

Uses CreateAction and UpdateAction class to model the contributions of *Context Entities* of type Person or Organization in the creation of files.

In this example the CreateAction has a human agent, the object is a Place (a cave) and the Hovermap drone is the instrument used in the file creation event.

```
{
      "@id": "#DataCapture_wcc02",
      "@type": "CreateAction",
      "agent": {
        "@id": "https://orcid.org/0000-0002-1672-552X"
      },
      "instrument": {
        "@id": "https://confluence.csiro.au/display/ASL/Hovermap"
      },
      "object": {
        "@id": "#victoria_arch"
      },
      "result": [
        {
          "@id": "wcc02_arch.laz"
        },
        {
          "@id": "wcc02_arch_traj.txt"
        }
      ]
    },
  {
      "@id": "#victoria_arch",
      "@type": "Place",
```

```
      "address": "Wombeyan Caves, NSW 2580",
      "name": "Victoria Arch"
  }
```

1.4.9 Provenance: Software used to create files

To specify which software was used to create or update a file the software application SHOULD be represented with an entity of type SoftwareApplication, with a version property, e.g. from `tool --version`.

For example:

```
{
      "@id": "https://www.imagemagick.org/",
      "@type": "SoftwareApplication",
      "url": "https://www.imagemagick.org/",
      "name": "ImageMagick",
      "version": "ImageMagick 6.9.7-4 Q16 x86_64 20170114 http://www.imagemagick.org"
}
```

The software SHOULD be associated with the File it created using a CreateAction with the File referenced by a result property. Any input files SHOULD be referenced by the object property.

In the below example, an image with the `@id` of `pics/2017-06-11%2012.56.14.jpg` was transformed into an new image `pics/sepia_fence.jpg` using the *ImageMagick* software application. Actions MAY have human-readable names, which MAY be machine generated for use at scale.

```
{
      "@id": "#Photo_Capture_1",
      "@type": "CreateAction",
      "agent": {
        "@id": "https://orcid.org/0000-0002-3545-944X"
      },
      "description": "Photo snapped on a photo walk on a misty day",
      "endTime": "2017-06-11T12:56:14+10:00",
      "instrument": [
        {
          "@id": "#EPL1"
        },
        {
          "@id": "#Panny20mm"
        }
      ],
      "result": {
        "@id": "pics/2017-06-11%2012.56.14.jpg"
```

```
    }
  },
  {
    "@id": "#SepiaConversion_1",
    "@type": "CreateAction",
    "name": "Convert dog image to sepia",
    "description": "convert -sepia-tone 80% test_data/sample/pics/2017-06-11\\ 12.56.14.jp
    "endTime": "2018-09-19T17:01:07+10:00",
    "instrument": {
      "@id": "https://www.imagemagick.org/"
    },
    "object": {
      "@id": "pics/2017-06-11%2012.56.14.jpg"
    },
    "result": {
      "@id": "pics/sepia_fence.jpg"
    }
  },
```

1.4.10 Provenance: Changes to RO-Crates

To record an action which changes the DataSet's metadata, or changes its state in a publication or other workflow, a CreateAction or UpdateAction SHOULD be associated with a *Data Entity*.

A curation Action MUST have at least one object which associates it with either the DataSet or one of its components.

An Action which creates new *Data entities* - for example, the creation of a new metadata file - SHOULD have these as results.

An Action SHOULD have a name and MAY have a description.

An Action SHOULD have an endTime, which MUST be in ISO 8601 date format and SHOULD be specified to at least the precision of a day. An Action MAY have a startTime meeting the same specifications.

An Action SHOULD have a human agent who was responsible for authorizing the action, and MAY have an instrument which associates the action with a particular piece of software (for example, the content management system or data catalogue through which an update was approved) which SHOULD be of `@type` SoftwareApplication.

An Action's status MAY be recorded in an actionStatus property. The status must be one of the values enumerated by ActionStatusType: ActiveActionStatus, CompletedActionStatus, FailedActionStatus or PotentialActionStatus.

An Action which has failed MAY record any error information in an error property.

UpdateAction SHOULD only be used for actions which affect the DataSet as a whole, such as movement through a workflow.

To record curation actions which modify a File within a DataSet - for example, by correcting or enhancing metadata - the old version of the File SHOULD be retained, and a CreateAction added which has the original version as its object and the new version as its result.

```
{
    "@id": "#history-01",
    "@type": "CreateAction",
    "object": { "@id": "https://doi.org/10.5281/zenodo.1009240" },
    "name": "RO-Crate created",
    "endTime": "2018-08-31",
    "agent": { "@id": "https://orcid.org/0000-0001-5152-5307" },
    "instrument": { "@id": "https://stash.research.uts.edu.au" },
    "actionStatus":  { "@id": "http://schema.org/CompletedActionStatus" }
},

{
    "@id": "#history-02",
    "@type": "UpdateAction",
    "object": { "@id": "https://doi.org/10.5281/zenodo.1009240" },
    "name": "RO-Crate published",
    "endTime": "2018-09-10",
    "agent": { "@id": "https://orcid.org/0000-0001-5152-5307" },
    "instrument": { "@id": "https://stash.research.uts.edu.au" },
    "actionStatus":  {"@id":" http://schema.org/CompletedActionStatus" }
},

{
    "@id": "#history-03",
    "@type": "CreateAction",
    "object": { "@id": "metadata.xml.v0.1" },
    "result": { "@id": "metadata.xml" },
    "name": "metadata update",
    "endTime": "2018-09-12",
    "agent": { "@id": "https://orcid.org/0000-0001-5152-5307" },
    "instrument": { "@id": "https://stash.research.uts.edu.au" },
    "actionStatus": { "@id": "http://schema.org/CompletedActionStatus" }
},

{
    "@id": "#history-04",
```

```
    "@type": "UpdateAction",
    "object": { "@id": "https://doi.org/10.5281/zenodo.1009240" },
    "name": "RO-Crate published",
    "endTime": "2018-09-13",
    "agent": { "@id": "https://orcid.org/0000-0001-5152-5307" },
    "instrument": { "@id": "https://stash.research.uts.edu.au" },
    "actionStatus": { "@id": "http://schema.org/FailedActionStatus" },
    "error": "Record is already published"
},


{
    "@id": "https://stash.research.uts.edu.au",
    "@type": "IndividualProduct",
    "name": "Stash",
    "description": "UTS Research Data Catalogue",
    "identifier": "https://stash.research.uts.edu.au"
}
```

1.4.11 Workflows and scripts

Scientific workflows and scripts that were used (or can be used) to analyze or generate files contained in an the RO-Crate MAY be embedded in an RO-Crate. Workflows and scripts SHOULD be described using the data entities of type SoftwareSourceCode.

The distinction between SoftwareSourceCode and SoftwareApplication for software is fluid, and comes down to availability and understandability. For instance, office spreadsheet applications are generally available and do not need further explanation (`SoftwareApplication`); while a Python script that is customized for a particular data analysis might be important to understand further and should therefore be included as `SoftwareSourceCode` in the RO-Crate dataset.

A workflow is a *Data Entity* which MUST have the following properties:

- `@type` is an array with at least `File` and `Workflow` as values.
- `@id` is a File URI linking to the workflow entry-point.
- name: a name for the workflow.

Minimal example describing a workflow:

```
{
    "@id": "workflow/retropath.knime",
    "@type": ["File", "SoftwareSourceCode", "Workflow"],
    "author": {"@id": "#thomas"},
    "name": "RetroPath Knime workflow",
    "description": "Retrosynthesis workflow calculating chemical reactions",
```

```
    "license": { "@id": "https://spdx.org/licenses/CC-BY-NC-SA-4.0"},
    "programmingLanguage": {"@id": "#knime"}
}
```

The `@type` property SHOULD be an array, to also indicate the particular nature of the source code as a *script* or a *workflow*. The distinction is fluid, depending on if the code is primarily indicating **what** should be done (`Workflow`), or **how** tasks should be executed (`Script`).

To indicate that a `SoftwareSourceCode` is primarily in the form of an executable **script** (e.g. sequential batch/shell script that call other commands and manage files), use:

```
  "@type": ["File", "SoftwareSourceCode", "Script"],
```

If the `SoftwareSourceCode` is primarily in the form of a **workflow** (e.g. a pipeline of steps with data flow), use:

```
  "@type": ["File", "SoftwareSourceCode", "Workflow"],
```

Workflows and scripts saved on disk using a *programming language* generally need a *runtime*, in RO-Crate this SHOULD be indicated using a liberal interpretation of programmingLanguage.

Note that the language and its runtime MAY differ (e.g. multiple different C++-compilers), but for scripts and workflows, frequently the language and runtime are essentially the same, and thus the `programmingLanguage` can be described in one go as a hybrid of a ComputerLanguage and SoftwareApplication:

```
{
  "@id": "scripts/analyse_csv.py",
  "@type": ["File", "SoftwareSourceCode", "Script"],
  "name": "Analyze CSV files",
  "programmingLanguage": {"@id": "https://www.python.org/downloads/release/python-380/"},
},
{
  "@id": "https://www.python.org/downloads/release/python-380/",
  "@type": ["ComputerLanguage", "SoftwareApplication"],
  "name": "Python 3.8.0",
  "version": "3.8.0"
}
```

A *contextual entity* representing a SoftwareApplication or ComputerLanguage MUST have a name, url and version, which should indicate a known version the workflow/script was developed or tested with. alternateName MAY be provided if there is a shorter colloquial name, for instance *"R"* instead of *"The R Project for Statistical Computing"*.

It is possible to indicate *steps* that are executed as part of an `Workflow` or `Script`, by using hasPart to relate additional `SoftwareApplication` or nested `SoftwareSourceCode` contextual entities:

```
{
    "@id": "workflow/analyze.cwl",
    "@type": ["File", "SoftwareSourceCode", "Workflow"],
    "name": "CWL workflow to analyze CSV and make PNG",
    "programmingLanguage": {"@id": "https://w3id.org/cwl/v1.1/"},
    "hasPart": [
      {"@id": "scripts/analyse_csv.py"},
      {"@id": "https://www.imagemagick.org/"},
    ]
}
```

1.4.11.1 Workflow diagram/sketch

It can be beneficial to show a diagram or sketch to explain the script/workflow. This may have been generated from a workflow management system, or drawn manually as a diagram. This diagram MAY be included as an ImageObject which is about the `SoftwareSourceCode`. The `@type` parameter SHOULD be an array to also include `WorkflowSketch` to indicate that this is an image that represent a sketch or diagram of the workflow.

```
{
  "@id": "workflow/workflow.svg",
  "@type": ["File", "ImageObject", "WorkflowSketch"],
  "encodingFormat":  "image/svg+xml",
  "name": "Diagram of RetroPath2.0 workflow",
  "about": {"@id": "workflow/workflow.knime"}
}
```

The image file format SHOULD be indicated with encodingFormat using an IANA registered media type like `image/svg+xml` or `image/png`. Additionally a reference to Pronom identifier SHOULD be provided, which MAY be described as an additional contextual entity to give human-readable name to the format:

```
{
  "@id": "workflow/workflow.svg",
  "@type": ["File", "ImageObject", "WorkflowSketch"],
  "encodingFormat":  ["image/svg+xml"],
  "description": "Diagram of RetroPath2.0 workflow",
  "about": {"@id": "workflow/workflow.knime"}
},
```

A *Sketch* may still be provided even if there is no programmatic `SoftwareSourceCode` that can be executed (e.g. because the workflow was done by hand). In this case the sketch itself is a proxy for the workflow and SHOULD have `@type` of `Workflow` and an `about` property referring to the *RO-Crate dataset* as a whole (assuming the RO-Crate represents the outcome of a single workflow), or to other `Data Entities` otherwise:

```
{
  "@id": "workflow/workflow.svg",
  "@type": ["File", "ImageObject", "WorkflowSketch", "Workflow"],
  "encodingFormat":  ["image/svg+xml"],
  "name": "Diagram of an ad hoc workflow",
  "about": {"@id": "./"}
}
```

1.4.12 Extra metadata such as Exif

Schema.org has a generic extension mechanism for encoding adding arbitrary properties and values which are not available as Schema.org properties. An example of of this is the Schema.org recommended way (see example 2) of including Exif technical image metadata.

To include EXIF, or other data which can be encoded as property/value pairs, add an array of references to *Anonymous Entities* which encode each property. This example shows one property of several hundred.

```
{
      "@id": "pics/2017-06-11%2012.56.14.jpg",
      "@type": ["File", "ImageObject"],
      "contentSize": "5114778",
      "author": {
        "@id": "https://orcid.org/0000-0002-3545-944X"
      },
      "description": "Depicts a fence at a disused motor racing venue with the front part of
      "encodingFormat": "image/jpeg",
      "exifData": [
        {
          "@id": "#2eb90b09-a8b8-4946-805b-8cba077a7137"
        },
        {
          "@id": "#c2521494-9b94-4b23-a713-6b281f540823"
        },
      ]

{
      "@id": "#c2521494-9b94-4b23-a713-6b281f540823",
      "@type": "PropertyValue",
      "name": "InternalSerialNumber",
      "value": "4102011002108002                "
    },
```

1.4.13 Places

To associate a *Data Entity* with a *Contextual Entity* representing a *geographical location or region* the entity SHOULD have a property of contentLocation with a value of type Place.

This example shows how to define a place, using a geonames ID:

```
{
  "@id": "http://sws.geonames.org/8152662/",
  "@type": "Place",
  "description": "Catalina Park is a disused motor racing venue, located at Katoomba ...",
  "geo": {
    "@id": "#b4168a98-8534-4c6d-a568-64a55157b656"
  },
  "identifier": "http://sws.geonames.org/8152662/",
  "uri": "https://www.geonames.org/8152662/catalina-park.html",
  "name": "Catalina Park"
},
```

Tip: To find the `@id` and `identifier` corresponding to a GeoNames HTML page like https://www.geonames.org/8152662/catalina-park.html click it's `.rdf` button to find the identifier http://sws.geonames.org/8152662/ referred from https://sws.geonames.org/8152662/about.rdf:

```
<gn:Feature rdf:about="http://sws.geonames.org/8152662/">
<!--... -->
```

The place has a geo property, referencing an *Contextual Entity* of `@type` GeoCoordinates:

```
{
  "@id": "#b4168a98-8534-4c6d-a568-64a55157b656",
  "@type": "GeoCoordinates",
  "latitude": "-33.7152",
  "longitude": "150.30119",
  "name": "Latitude: -33.7152 Longitude: 150.30119"
},
```

The GeoCoordinates item SHOULD have a human readable name, which is used in generating the `ro-crate-preview.html` file.

And the place is referenced from the contentLocation property of the dataset.

```
{
  "@id": "./",
  "@type": "Dataset",
  "outputOf": "RO-Crate",
  "contact": {
    "@id": "https://orcid.org/0000-0002-3545-944X"
```

```
  },
  "contentLocation": {
    "@id": "http://sws.geonames.org/8152662/",
  }
}
{
  "@id": "http://sws.geonames.org/8152662/",
  "name": "Catalina Park",
}
```

Place MAY use any of the resources available in Schema.org to describe places. Future profiles of RO-Crate may mandate the use of a subset of these. Any directory or file or *Contextual Entity* may be geo-located. For example this file:

```
{
  "@id": "pics/19093074_10155469333581584_5707039334816454031_o.jpg",
  "@type": "File",
  "contentLocation": {
    "@id": "http://sws.geonames.org/8152662/"
  },
  "contentSize": "132765",
  "author": {
    "@id": "https://orcid.org/0000-0002-3545-944X"
  },
```

1.4.14 Subjects & keywords

Subject properties (equivalent to a Dublin Core Subject) on RO-Crate or a data entity MUST use the about property.

Keyword properties MUST use keywords. Note that by schema.org convention, keywords are given as a single JSON string, with individual keywords separated by commas.

```
{
  "keywords": "Gibraltar, Spain, British Overseas Territory, city, map",
  "about": { "@id": "http://dbpedia.org/resource/Gibraltar" },
}
```

1.4.15 Time

To describe the time period which a RO-Crate Data Entity (or the RO-Crate itself) is *about*, use temporalCoverage:

```
{
  "@id": "photos/",
  "@type": "Dataset",
```

```
  "name": "Photos of Gibraltar from 1950 till 1975",
  "about": {"@id": "http://dbpedia.org/resource/Gibraltar"},
  "temporalCoverage": "1950/1975"
}
```

1.4.16 Digital Library and Repository content

To describe an export from a Digital Library or repository system, RO-Crate uses the *Portland Common Data Model* (PCDM). A *Contextual Entity* from a repository, representing an abstract entity such as a person, or a work, or a place SHOULD have a`@type` of RepositoryObject, in addition to any other types. Objects MAY be grouped together in RepositoryCollections with hasMember pointing to the the RepositoryObject. The keys RepositoryObject and RepositoryCollection were chosen to avoid collision between the terms Collection and Object with other vocabularies.

NOTE: PCDM specifies that Files should have only technical metadata, not descriptive metadata, which is *not* a restriction in RO-Crate. If the RO-Crate is to be imported into a strict PCDM repository, modeling of object/file relationships will be necessary.

For example, this data is exported from an Omeka repository:

```
{
   "@id": "https://omeka.uws.edu.au/farmstofreeways/api/collections/6",
   "@type": "RepositoryCollection",
   "title":  "Project Materials",
   "description": [
      "Materials associated with the project, including fliers seeking participants, lists o
   ],
   "publisher": {"@id": "University of Western Sydney"},
   "rights": "Copyright University of Western Sydney 2015",
   "hasMember": [
      {
         "@id": "https://omeka.uws.edu.au/farmstofreeways/api/items/166"
      },
      {
         "@id": "https://omeka.uws.edu.au/farmstofreeways/api/items/167"
      },
      {
         "@id": "https://omeka.uws.edu.au/farmstofreeways/api/items/168"
      },
      {
         "@id": "https://omeka.uws.edu.au/farmstofreeways/api/items/169"
      }
   ]
```

```
},
{
   "@id": "https://omeka.uws.edu.au/farmstofreeways/api/items/166",
   "@type": "RepositoryObject",
   "title": [
      "Western Sydney Women's Oral History Project: Flier (illustrated)"
   ],
   "description": [
      "Flier (illustrated) seeking participants for the project."
   ],
   "publisher": { "@id": "https://westernsydney.edu.au"},
   "rights": "Copyright University of Western Sydney 2015",
   "originalFormat": "Paper",
   "identifier": "FTF_flier_illust"
   ],
   "rightsHolder": [
      "Western Sydney University"
   ],
   "license": {
     "@id": "https://creativecommons.org/licenses/by/3.0/au/"
   },
   "hasFile": [
      {
         "@id": "content/166/original_eece70f73bf8979c0bcfb97065948531.pdf"
      },
     ...
   ]
},
{
   "@type": "File",
   "@id": "content/166/original_eece70f73bf8979c0bcfb97065948531.pdf"
}
```

1.4.17 Thumbnails

A File or any other item MAY have a thumbnail property which references another file.

For example, the below RepositoryObject is related to four files which are all versions of the same image (via hasFile) one of which is a thumbnail. The thumbnail MUST be included in the RO-Crate.

If thumbnails are incidental to the data set, they need not be referenced by hasPart or hasFile relationships. but must be in the BagIt manifest if in a *Bagged RO-Crate*.

```
{
  "@id": "https://omeka.uws.edu.au/farmstofreeways/api/items/383",
  "@type": [
    "RepositoryObject",
    "Image"
  ],
  "identifier": [
    "ftf_photo_stapleton1"
  ],
  "interviewee": [
    {
      "@id": "https://omeka.uws.edu.au/farmstofreeways/api/items/595",
    }
  ],
  "description": [
    "Photo of Eugenie Stapleton inside her home"
  ],
  "license": [
    "Content in the Western Sydney Women's Oral History Project: From farms to freeways coll
  ],
  "publisher": [
    "University of Western Sydney"
  ],
  "hasFile": [
    {
      "@id": "files/383/original_c0f1189ec13ca936e8f556161663d4ba.jpg"
    },
    {
      "@id": "files/383/fullsize_c0f1189ec13ca936e8f556161663d4ba.jpg"
    },
    {
      "@id": "files/383/thumbnail_c0f1189ec13ca936e8f556161663d4ba.jpg"
    },
    {
      "@id": "files/383/square_thumbnail_c0f1189ec13ca936e8f556161663d4ba.jpg"
    }
  ],
  "thumbnail": [
    {
      "@id": "files/383/thumbnail_c0f1189ec13ca936e8f556161663d4ba.jpg"
    }
  ],
  "name": [
    "Photo of Eugenie Stapleton 1"
  ],
  "relatedLink": [
```

```
    "<a href=\"https://omeka.uws.edu.au/farmstofreeways/items/show/512\">Audio recording of
  ],
  "copyrightHolder": [
    { "@id": "https://westernsydney.edu.au"}
  ],
  "copyright": [
    "Copyright University of Western Sydney 2015"
  ]
},
{
  "@type": "File",
  "@id": "files/384/original_2ebbe681aa6ec138776343974ce8a3dd.jpg"
},
{
  "@type": "File",
  "@id": "files/384/fullsize_2ebbe681aa6ec138776343974ce8a3dd.jpg"
},
{
  "@type": "File",
  "@id": "files/384/thumbnail_2ebbe681aa6ec138776343974ce8a3dd.jpg"
},
{
  "@type": "File",
  "@id": "files/384/square_thumbnail_2ebbe681aa6ec138776343974ce8a3dd.jpg"
},
```

1.5 APPENDIX: RO-Crate JSON-LD

It is not necessary to use JSON-LD tooling to generate or parse the *RO-Crate Metadata File*, although they may make it easier to conform to this specification, e.g. handling relative URIs. It is RECOMMENDED to use JSON tooling to handle JSON syntax and escaping rules.

This appendix shows a brief JSON-LD introduction for complying with the RO-Crate Metadata File requirements.

The below example shows the overall structure of a flattened, compacted `ro-crate-metadata.jsonld` where `@context` refers to the *RO-Crate JSON-LD Context*, while `@graph` is a flat list of the entities that constitute this RO-Crate.

```
{ "@context": "https://w3id.org/ro/crate/1.0/context",
  "@graph": [

    {
```

```
    "@type": "CreativeWork",
    "@id": "ro-crate-metadata.jsonld",
    "conformsTo": {"@id": "https://w3id.org/ro/crate/1.0"},
    "about": {"@id": "./"},
    "description": "RO-Crate Metadata File Descriptor (this file)"
  },
  {
    "@id": "./",
    "@type": "Dataset",
    "name": "Example RO-Crate",
    "description": "The RO-Crate Root Data Entity",
    "hasPart": [
      {"@id": "data1.txt"},
      {"@id": "data2.txt"}
    ]
  },


  {
    "@id": "data1.txt",
    "@type": "File",
    "description": "One of hopefully many Data Entities",
    "author": {"@id": "#alice"},
    "contentLocation":  {"@id": "http://sws.geonames.org/8152662/"}
  },
  {
    "@id": "data2.txt",
    "@type": "File"
  },

  {
    "@id": "#alice",
    "@type": "Person",
    "name": "Alice",
    "description": "One of hopefully many Contextual Entities"
  },
  {
    "@id": "http://sws.geonames.org/8152662/",
    "@type": "Place",
    "name": "Catalina Park"
  }
 ]
}
```

Note: entities above have been shortened for brevity, see their individual sections elsewhere in this specification.

The order of the `@graph` list is not significant. Above we see that the RO-Crate JSON-LD graph contain the *RO-Crate Metadata File Descriptor*, the *Root Data Entity*, any *Data Entities* and any *Contextual Entities*.

1.5.1 Describing entities in JSON-LD

Properties of an entity can refer to another URL or entity by using the form `{"@id": "uri-reference"}` as in the example above, where the author property in the File entity refer to the Person entity, identified as `#alice`.

Identifiers in `@id` SHOULD be either a valid *absolute URIs* like http://example.com/, or an *URI references URI paths* relative to the RO-Crate root directory. Care must be taken to express any relative paths using `/` separator and escape special characters like space (`%20`). As JSON-LD supports *IRIs*, international characters in identifiers SHOULD be encoded in UTF-8 rather than `%`-escaped.

Because the *RO-Crate JSON-LD* is *flattened*, all described entities must be direct children of the `@graph` element rather than being nested under another property or list.

If no obvious identifier is available for a contextual entity, an identifier local to the *RO-Crate Metadata File* can be generated, for instance `{"@id": "#alice"}` or `{"@id": "#ac0bd781-7d91-4cdf-b2ad-7305921c7650"}`. Although it is RECOMMENDED to use `#`-based local identifiers, identifiers in `@id` MAY alternatively be a *blank node* identifier (e.g. `_:alice`).

Multiple values and references can be represented using JSON arrays, as exemplified in `hasPart` above, however as the `RO-Crate JSON-LD` is in *compacted form* any single-element arrays like `"author": [{"@id": "#alice"}]` SHOULD be unpacked to a single value like `"author": {"@id": "#alice"}`.

1.5.2 RO-Crate JSON-LD Context

The main purpose of the `@context` is to relate JSON property keys and `@type` references to their Linked Data identifiers, which in RO-Crate is based primarily on http://schema.org/ URIs.

In other uses of JSON-LD the context may perform more automatic or detailed mapping, but the `RO-Crate JSON-LD context` is deliberately flat, listing every property and type.

To find the full description of a particular property or type, follow its URI from the context. For instance, we can find within the context https://w3id.org/ro/crate/1.0/context that `author` above is mapped to http://schema.org/author:

```
"author": "http://schema.org/author",
```

The *RO-Crate JSON-LD Context* may either be set by reference to https://w3id.org/ro/crate/1.0/context, or by value (merging the two documents).

Consider the below (simplified) example of *by reference* using a versioned permalink:

```
{ "@context": "https://w3id.org/ro/crate/1.0/context",
  "@graph": [
    {
      "@id": "ro-crate-metadata.jsonld",
      "@type": "CreativeWork",
      "description": "RO-Crate Metadata File Descriptor (this file)",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.0"},
      "about": {"@id": "./"}
    }
  ]
}
```

The above is equivalent to this JSON-LD using an embedded context, by adding the subset of corresponding keys from the external `@context`:

```
{ "@context": {
      "CreativeWork": "http://schema.org/CreativeWork",
      "about": "http://schema.org/about",
      "description": "http://schema.org/description",
      "conformsTo": "http://purl.org/dc/terms/conformsTo",
      "about": "http://schema.org/about"
  },
  "@graph": [
    {
      "@id": "ro-crate-metadata.jsonld",
      "@type": "CreativeWork",
      "description": "RO-Crate Metadata File Descriptor (this file)",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.0"},
      "about": {"@id": "./"}
    }
  ]
}
```

While the second form is more verbose, one advantage is that it is "archivable" as it does not require Internet access for retrieving the `@context` permalink. Tools consuming or archiving RO-Crate MAY replace by-reference `@context` URIs with an embedded context by using version-specific hard-coded contexts, see https://github.com/ResearchObject/ro-crate/releases

To check which RO-Crate version is used (in terms of properties and types expected), clients SHOULD check the property `conformsTo` on the *RO-Crate Metadata File Descriptor* rather than the value of `@context`.

RO-Crate consumers SHOULD NOT do the opposite substitution from an embedded context, but MAY use the JSON-LD flattening algorithm with *compaction* to a referenced *RO-Crate JSON-LD context*.

Tip: The JSON-LD flattening & compaction algorithms can be used to rewrite to a different `@context`, e.g. to http://schema.org or a different version of the *RO-Crate JSON-LD Context*.

1.5.3 RO-Crate JSON-LD Media type

The media type for `ro-crate-metadata.jsonld` will, when following this specification, comply with the flattened/compacted JSON-LD profiles as well as `https://w3id.org/ro/crate`, which may be indicated in a HTTP response as:

```
HEAD http://example.com/ro-123/ro-crate-metadata.jsonld HTTP/1.1

HTTP/1.1 200 OK
Content-Type: application/ld+json; profile="http://www.w3.org/ns/json-ld#flattened http
```

1.5.4 Extending RO-Crate

To extend RO-Crate, implementers SHOULD try to use existing http://schema.org/ properties and classes and MAY use terms from other vocabularies and ontologies when this is not possible.

The terms (properties and classes) used SHOULD be added as keys to the `@context` in the *RO-Crate JSON-LD* (if not present). To avoid duplicating the *RO-Crate JSON-LD Context* the `@context: []` array form SHOULD be used as shown below.

URIs in the `@context` SHOULD resolve to a useful human readable page. Where this is not possible - for example if the URI resolves to an RDF ontology file, a human-readable URI SHOULD be provided using a sameAs description.

For example. The `@id` URI http://purl.org/ontology/bibo/interviewee from the BIBO ontology ontology itends to resolve to an ontology file, which is not useful for humans, however the HTML section http://neologism.ecs.soton.ac.uk/bibo.html#interviewee is human-readable.

```
{
  "@context": [
    "https://w3id.org/ro/crate/1.0/context",
    {"interviewee": "http://purl.org/ontology/bibo/interviewee"},
  ],
  "@graph": [
  {
      "@id": "http://purl.org/ontology/bibo/interviewee",
```

```
      "sameAs": "http://neologism.ecs.soton.ac.uk/bibo.html#interviewee",
      "@type": "Thing"
  }
 ]
}
```

When generating the *RO-Crate Website* from *RO-Crate JSON-LD*, the code MUST use a sameAs URI (if present) as a target for an explanatory link for the term instead of the Linked Data URI supplied in the `@context`.

Where there is no RDF ontology available, then implementors SHOULD attempt to provide context by creating stable web-accessible URIs to document properties and classes, for example, by linking to page describing an XML element or attribute from an XML schema, pending the publication of a formal ontology.

1.6 APPENDIX: Implementation notes

1.6.0.1 Combining with other packaging schemes RO-Crates may co-exist with other packaging schemes, such as BagIt using two general approaches; either (a) *adding* RO-Crate into a package as part of the payload or (b) *wrapping* another kind of package. Examples using BagIt follow.

BagIt is described in RFC 8493:

> [BagIt is] ... a set of hierarchical file layout conventions for storage and transfer of arbitrary digital content. A "bag" has just enough structure to enclose descriptive metadata "tags" and a file "payload" but does not require knowledge of the payload's internal semantics. This BagIt format is suitable for reliable storage and transfer.

BagIt and RO-Crate have largely separate concerns - RO-Crate is focussed on rich metadata, the semantics of data, while BagIt is about reliable transfer.

1.6.0.1.1 Example of adding RO-Crate to Bagit RO-Crate can be combined with BagIt simply by placing the RO-Crate files in the BagIt payload (`data/`) directory.

```
<RO-Crate root directory>/
  |   bagit.txt                # As per BagIt specification
  |   bag-info.txt             # As per BagIt specification
  |   manifest-<algorithm>.txt  # As per BagIt specification
  |   fetch.txt                # Optional, per BagIt Specification
  |   data/
      |   ro-crate-metadata.jsonld  # RO-Crate Metadata File MUST be present
      |   ro-crate-preview.html     # RO-Crate Website homepage MAY be present
      |   ro-crate-preview_files/   # MAY be present
      |     [payload files and directories]  # 1 or more SHOULD be present
```

1.6.0.1.2 Example of wrapping a BagIt bag in an RO-Crate

Alternatively, an RO-Crate can wrap a BagIt bag, so that the RO-Crate metadata is outside of the bag directory and can be changed without changing the payload's checksums.

```
<RO-Crate root directory>/
  |   ro-crate-metadata.jsonld  # RO-Crate Metadata File MUST be present
  |   ro-crate-preview.html     # RO-Crate Website homepage MAY be present
  |   ro-crate-preview_files/   # MAY be present
  |   bag/                      # "Wrapped" bag - could have any name
  |      bagit.txt                 # As per BagIt specification
  |      bag-info.txt              # As per BagIt specification
  |      manifest-<algorithm>.txt  # As per BagIt specification
  |      fetch.txt                 # Optional, per BagIt Specification
  |      data/
  |         [payload files and directories]  # 1 or more SHOULD be present
  |          example.txt
```

A *Data Entity* describing example.txt would have an `@id` of `bag/data/example.txt`:

```
{
  "@id": "bag/data/example.txt",
  "name": "Example file"
}
```

1.6.0.2 Repository-specific identifiers

Root Data Entities MAY also have additional repository specific identifiers, described using `Contextual Entities` using a PropertyValue, with a `name` that identifies the repository and the `identifier` as a value. The *same* identifier MAY be used in multiple different repositories and effectively namespaced using the `name` of the ProperyValue.

```
{
  "@id": "./",
  "@type": "Dataset",
  "identifier": ["https://doi.org/10.4225/59/59672c09f4a4b", {"@id": "_:localid:my-repo:my-i
}

 {
   "@id": "_:localid:my-repo:my-id",
   "@type": "PropertyValue",
   "name": "my-repo",
   "value": "my-id"
 }
```

```
 {
  "@id": "_:localid:other-repo:https://doi.org/10.4225/59/59672c09f4a4b",
  "@type": "PropertyValue",
  "name": "other-repo",
  "value": "https://doi.org/10.4225/59/59672c09f4a4b"
}
```