Contents

1 RO-Crate Metadata Specification 1.1 **3**

2 Introduction **4**

3 Terminology **4**
3.1 Linked Data conventions 5

4 RO-Crate Structure **5**
4.1 RO-Crate Metadata File (`ro-crate-metadata.json`) 6
4.2 RO-Crate Website (`ro-crate-preview.html` and `ro-crate-preview_files/`) 7
4.3 Payload files and directories 7
4.4 Self-describing and self-contained 8

5 RO-Crate Metadata **8**
5.1 RO-Crate uses Linked Data principles 9
5.2 Base metadata standard: Schema.org 9
5.2.1 Differences from Schema.org 10
5.3 Additional metadata standards 10
5.4 Summary of Coverage 11
5.5 Future coverage 11
5.6 Recommended Identifiers 12

6 Root Data Entity **12**
6.1 RO-Crate Metadata File Descriptor 12
6.1.1 Finding the Root Data Entity 13
6.1.2 Purpose of Metadata File 13
6.2 Direct properties of the Root Data Entity 13
6.3 Minimal example of RO-Crate 14

7 Data Entities **14**
7.1 Referencing files and folders from the Root Data Entity 15
7.1.1 Example linking to a file and folders 15
7.1.2 Adding detailed descriptions of encodings 16
7.2 Core Metadata for Data Entities 17
7.2.1 Encoding file paths 17
7.2.2 File Data Entity 18
7.2.3 Directory File Entity 18
7.3 Web-based Data Entities 18
7.3.1 Embedded data entities that are also on the web 20
7.3.2 Directories on the web; dataset distributions 20

8 Representing Contextual Entities **21**
8.1 Contextual vs Data entities 21
8.2 Identifiers for contextual entities 22
8.3 People 22
8.4 Organizations as values 22
8.5 Contact information 23
8.6 Publications via citation property 24
8.7 Publisher 25

8.8 Funding and grants . . . 25
8.9 Licensing, Access control and copyright . . . 26
8.9.1 Metadata license . . . 27
8.10 Places . . . 29
8.11 Subjects & keywords . . . 31
8.12 Time . . . 31
8.13 Thumbnails . . . 31

9 Detailing provenance of entities 33
9.1 Equipment used to create files . . . 33
9.2 Software used to create files . . . 34
9.3 Recording changes to RO-Crates . . . 35
9.4 Digital Library and Repository content . . . 37

10 Workflows and Scripts 39
10.1 Describing scripts and workflows . . . 39
10.2 Workflow Runtime and Programming Language . . . 40
10.3 Workflow diagram/sketch . . . 41
10.4 Complying with Bioschemas Computational Workflow profile . . . 42
10.4.1 Describing inputs and outputs . . . 42
10.5 Complete Workflow Example . . . 43
10.6 Appendixes . . . 45

11 APPENDIX: Changelog 45

12 APPENDIX: Implementation notes 47
12.1 Programming with JSON-LD . . . 47
12.2 Combining with other packaging schemes . . . 47
12.2.1 Adding RO-Crate to Bagit . . . 48
12.2.2 Example of wrapping a BagIt bag in an RO-Crate . . . 50
12.3 Repository-specific identifiers . . . 51

13 APPENDIX: RO-Crate JSON-LD 51
13.1 Describing entities in JSON-LD . . . 53
13.2 RO-Crate JSON-LD Context . . . 53
13.3 RO-Crate JSON-LD Media type . . . 55
13.4 Extending RO-Crate . . . 55
13.5 Adding new or ad hoc vocabulary terms . . . 56
13.5.1 Choosing URLs for ad hoc terms . . . 56
13.5.2 Add local definitions of ad hoc terms . . . 57

14 APPENDIX: Handling relative URI references 58
14.1 Flattening JSON-LD from nested JSON . . . 58
14.2 Expanding/parsing JSON-LD keeping relative referencing . . . 60
14.3 Establishing absolute URI for RO-Crate Root . . . 62
14.4 Finding RO-Crate Root in RDF triple stores . . . 63
14.5 Parsing as RDF with a different RO-Crate Root . . . 64
14.6 Establishing a base URI inside a ZIP file . . . 66
14.7 Relativizing absolute URIs within RO-Crate Root . . . 67

15 References 69

1 RO-Crate Metadata Specification 1.1

- Permalink: https://w3id.org/ro/crate/1.1
- Published: 2020-10-30
- Publisher: researchobject.org community
- Status: Recommendation
- JSON-LD context: https://w3id.org/ro/crate/1.1/context
- This version: https://w3id.org/ro/crate/1.1
- Alternate formats: Web pages, single-page HTML, PDF, RO-Crate JSON-LD, RO-Crate HTML
- Previous version: https://w3id.org/ro/crate/1.0
- Cite as: https://doi.org/10.5281/zenodo.4031327 (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, Hervé Ménager, Laura Rodríguez Navas, Paul Walk, brandon whitehead, Mark Wilkinson, Paul Groth, Erich Bremer, LJ Garcia Castro, Karl Sebby, Alexander Kanitz, Ana Trisovic, Gavin Kennedy, Mark Graves, Jasper Koehorst, Simone Leo

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

2 Introduction

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.json`. 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.

3 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.json` (or `ro-crate-metadata.jsonld` for crates that comply with versions before v1.1 of this specification)

RO-Crate Metadata File: A JSON-LD file stored as `ro-crate-metadata.json` in the *RO-Crate Root*. The metadata file describes the *RO-Crate* with structured data in form of *RO-Crate JSON-LD*. (In version 1.0 this file was named `ro-crate-metadata.jsonld` but has been renamed to improve the usability of crates.)

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/`)

Entity: An identified object, which have a given *type* and may be described using a set of *properties*.

Type: A classification of objects or their descriptions. The type (or *class*) is identified by a *URI*, mapped to a *key* by *JSON-LD*.

Property: A relationship from one *entity* to another entity, or to a *value*. The type of relationship is identified by a *URI*, mapped to a *key* by *JSON-LD*.

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. The main JSON structures are *objects* (`{}`) indexed by *keys*, sequential *arrays* (`[]`) and literal *values* (`""`).

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*.

3.1 Linked Data conventions

Throughout this specification, RDF terms (*properties*, *types*) 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 `Type` names start with uppercase letters.

In the *RO-Crate Metadata File* 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/1.1/context

4 RO-Crate Structure

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

```
<RO-Crate root directory>/
|   ro-crate-metadata.json    # 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]  # 0 or more
```

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

Data Entities in the RO-Crate MUST either be *payload files/directories* present within the RO-Crate root directory or its subdirectories, or be Web-based Data Entities.

4.1 RO-Crate Metadata File (`ro-crate-metadata.json`)

- In new RO-Crates the *RO-Crate Metadata File* MUST be named `ro-crate-metadata.json` 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/1.1/context by reference.
- If an RO-Crate conforming to version 1.0 or earlier contains a file named `ro-crate-metadata.jsonld` instead of `ro-crate-metadata.json` then processing software should treat this as the *RO-Crate Metadata File*. If the crate is updated then the file SHOULD be renamed to `ro-crate-metadata.json` and the *RO-Crate Metadata File Descriptor* SHOULD be updated to to reference it, with an up to date conformsTo property naming an appropriate version of this specification.

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` array below:

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
```

```
  ]
}
```

4.2 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
- Be useful to users of the RO-Crate - this will vary by community and intended use, but in general the aim to assist users in reusing data by explaining what it is, how it was created how it can be used and how to cite it. One simple approach to this is to expose *all* the metadata in the *RO-Crate Metadata File*.
- 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/1.1/context",
    "@graph": [ ...]
}
</script>
```

`ro-crate-preview.html` SHOULD:

- 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 external URI values, provide a link.
- 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).
- 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/`

4.3 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*.

4.4 Self-describing and self-contained

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

A minimal RO-Crate is a directory containing a single RO-Crate Metadata File `ro-crate-metadata.json`.

At the basic level, an RO-Crate is a collection of files and resources 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*.

While RO-Crate is well catered for describing a *Dataset* as files and relevant metadata that are *contained* by the RO-Crate in the sense of living within the same root directory, RO-Crates can also reference external resources which are stored or accessed separately, via absolute URIs. This is particularly recommended where some resources cannot be co-hosted for practical or legal reasons, or if the RO-Crate itself is primarily web-based.

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 (see also appendix Combining with other packaging schemes). An RO-Crate can also be hosted on the web or mainly refer to web resources, although extra care to ensure persistence and consistency should be taken for archiving such RO-Crates.

5 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 RO-Crate `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.

5.1 RO-Crate uses Linked Data principles

RO-Crate makes use of the 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-LD.
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.

5.2 Base metadata standard: 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.

5.2.1 Differences from Schema.org

Generally, the standard *type* and *property* names (*terms*) 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.

Warning

JSON-LD examples given on the [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.

5.3 Additional metadata standards

RO-Crate also uses the *Portland Common Data Model* (PCDM version https://pcdm.org/2016/04/18/models) to describe repositories or collections of digital objects and imports these terms:

- `RepositoryObject` mapped to http://pcdm.org/models#Object
- `RepositoryCollection` mapped to http://pcdm.org/models#Collection
- `RepositoryFile` mapped to http://pcdm.org/models#File
- `hasMember` mapped to http://pcdm.org/models#hasMember
- `hasFile` mapped to http://pcdm.org/models#hasFile

Note

The terms `RepositoryObject` and `RepositoryCollection` are renamed to avoid collision between other vocabularies and the PCDM terms `Collection` and `Object`. The term `RepositoryFile` is renamed to avoid clash with RO-Crate's `File` mapping to http://schema.org/MediaObject.

From Dublin Core Terms RO-Crate use:

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

These terms are being proposed by Bioschemas profile ComputationalWorkflow 0.5-DRAFT and FormalParameter 0.1-DRAFT to be integrated into Schema.org:

- `ComputationalWorkflow` mapped to https://bioschemas.org/ComputationalWorkflow
- `FormalParameter` mapped to https://bioschemas.org/FormalParameter

- `input` mapped to https://bioschemas.org/ComputationalWorkflow#input
- `output` mapped to https://bioschemas.org/ComputationalWorkflow#output
- `funding` mapped to http://schema.org/funding (schemaorg #383)

NOTE

In this specification the proposed Bioschemas terms use the temporary https://bioschemas.org/ namespace; future releases of RO-Crate may reflect mapping to the http://schema.org/ namespace.

5.4 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 two 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 the 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.

5.5 Future coverage

A future version of this specification aim to cater 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.

5.6 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 entities 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.

6 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*.

As explained in section RO-Crate structure, the RO-Crate description is stored as *JSON-LD* in the *RO-Crate Metadata File* `ro-crate-metadata.json` in the *RO-Crate root* directory.

6.1 RO-Crate Metadata File Descriptor

The *RO-Crate JSON-LD* MUST contain a self-describing **RO-Crate Metadata File Descriptor** with the `@id` value `ro-crate-metadata.json` (or `ro-crate-metadata.jsonld` in legacy crates) 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/1.1/context",
  "@graph": [
    {
        "@type": "CreativeWork",
        "@id": "ro-crate-metadata.json",
        "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
        "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/`.

6.1.1 Finding the Root Data Entity

Consumers processing the RO-Crate as an JSON-LD graph can thus reliably find 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

See also the appendix on finding RO-Crate Root in RDF triple stores.

6.1.2 Purpose of Metadata File

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 earlier 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 the appendix Extending RO-Crate).

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:

6.2 Direct properties of the Root Data Entity

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

- `@type`: MUST be Dataset
- `@id`: MUST end with `/` and SHOULD be the string `./`
- `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 Com-

mons 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`,

WARNING

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.

6.3 Minimal example of RO-Crate

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


```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [

 {
    "@type": "CreativeWork",
    "@id": "ro-crate-metadata.json",
    "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
    "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
    "description": "Palliative care planning for nursing home residents with advanced deme
    "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-NonCommerci
  "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)"
 }
 ]
}
```


7 Data Entities

The primary purpose for RO-Crate is to gather and describe a set of *Data entities* in the form of:

- Files
- Directories
- Web resources

The data entities can be further described by referencing contextual entitites such as persons, organizations and publications.

7.1 Referencing 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 have `"File"` as a value for `@type`. `File` is an RO-Crate alias for http://schema.org/MediaObject. The term *File* here is liberal, and includes "downloadable" resources where `@id` is an absolute URI.

Data Entities representing directories MUST be of `"@type": "Dataset"`. The term *directory* here includes HTTP file listings where `@id` is an absolute URI, however "external" directories SHOULD have a programmatic listing of their content (e.g. another RO-Crate).

Data Entities can also be other types, for instance an online database. These SHOULD be of `"@type": "CreativeWork"` and typically have a `@id` which is an absolute URI.

In all cases, `@type` MAY be an array in order to also specify a more specific type, e.g. `"@type": ["File", "ComputationalWorkflow"]`

Tip

There is no requirement to represent *every* file and folder in an RO-Crate as Data Entities in the RO-Crate JSON-LD.

7.1.1 Example linking to a file and folders

```
<RO-Crate root>/
  |   ro-crate-metadata.json
  |   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/1.1/context",
  "@graph": [
    {
      "@type": "CreativeWork",
```

```
      "@id": "ro-crate-metadata.json",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
      "about": {"@id": "./"}
    },
    {
      "@id": "./",
      "@type": [
        "Dataset"
      ],
      "hasPart": [
        {
          "@id": "cp7glop.ai"
        },
        {
          "@id": "lots_of_little_files/"
        }
      ]
    },
    {
      "@id": "cp7glop.ai",
      "@type": "File",
      "name": "Diagram showing trend to increase",
      "contentSize": "383766",
      "description": "Illustrator file for Glop Pot",
      "encodingFormat": "application/pdf"
    },
    {
      "@id": "lots_of_little_files/",
      "@type": "Dataset",
      "name": "Too many files",
      "description": "This directory contains many small files, that we're not going to de
    }
  ]
}
```

7.1.2 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 is readable as a PDF file from the extension alone. To add more detail, encodings SHOULD be linked using a PRONOM identifier to a *Contextual Entity* of `@type` WebSite.

```
  {
    "@id": "cp7glop.ai",
    "@type": "File",
    "name": "Diagram showing trend to increase",
    "contentSize": "383766",
    "description": "Illustrator file for Glop Pot",
    "encodingFormat": ["application/pdf", {"@id": "https://www.nationalarchives.gov.uk/PRO
  },
```

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

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",
   "name": "Some file",
   "description": "A file in a non-standard format",
   "contentSize": "120",
   "encodingFormat": ["text/plain", {"@id": "some_extension.md"}]
 },
 {
   "@id": "some_extension.md",
   "@type": ["File", "CreativeWork"],
   "name": "Description of some_extension file format",
   "encodingFormat": "text/markdown"
 }
```

7.2 Core Metadata for Data Entities

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

7.2.1 Encoding file paths

Note that all `@id` identifiers must be valid URI references, care must be taken to express any relative paths using `/` separator, correct casing, and escape special characters like space (`%20`) and percent (`%25`), for instance a *File Data*

Entity from the Windows path `Results and Diagrams\almost-50%.png` becomes `"@id": "Results%20and%20Diagrams/almost-50%25.png"` in the *RO-Crate JSON-LD*.

In this document the term *URI* includes international *IRI*s; the *RO-Crate Metadata File* is always UTF-8 and international characters in identifiers SHOULD be written using native UTF-8 characters (*IRI*s), however traditional URL encoding of Unicode characters with `%` MAY appear in `@id` strings. Example: `"@id": " .mp4"` is preferred over the equivalent `"@id": "%E9%9D%A2%E8%AF%95.mp4"`

7.2.2 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 either a *URI Path* relative to the *RO Crate root*, or an absolute URI.

7.2.3 Directory File Entity

A Dataset (directory) *Data Entity* MUST have the following properties:

- `@type` MUST be `Dataset` or an array where `Dataset` is one of the values.
- `@id` MUST be either an a *URI Path* relative to the *RO Crate root*, or an absolute URI. The id SHOULD end with `/`

7.3 Web-based Data Entities

While one use-case of RO-Crates is to describe *files* contained within the *RO-Crate root* directory, RO-Crates can also gather resources from the web identified by *absolute URIs* instead of relative *URI paths*, i.e. Web-based data entities.

Using Web-based data entities can be important particularly where a file can't be included in the *RO-Crate root* because of licensing concerns, large data sizes, privacy, or where it is desirable to link to the latest online version.

Example of an RO-Crate including a *File Data Entity* external to the *RO-Crate root* (file entity https://zenodo.org/record/3541888/files/ro-crate-1.0.0.pdf):

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
    {
        "@type": "CreativeWork",
        "@id": "ro-crate-metadata.json",
        "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
        "about": {"@id": "./"}
  },
  {
    "@id": "./",
    "@type": [
      "Dataset"
    ],
    "hasPart": [
```

```
      {
        "@id": "survey-responses-2019.csv"
      },
      {
        "@id": "https://zenodo.org/record/3541888/files/ro-crate-1.0.0.pdf"
      },
      ]
  },
  {
    "@id": "survey-responses-2019.csv",
    "@type": "File",
    "name": "Survey responses",
    "contentSize": "26452",
    "encodingFormat": "text/csv"
  },
  {
    "@id": "https://zenodo.org/record/3541888/files/ro-crate-1.0.0.pdf",
    "@type": "File",
    "name": "RO-Crate specification",
    "contentSize": "310691",
    "description": "RO-Crate specification",
    "encodingFormat": "application/pdf"
  }
]
}
```

Additional care SHOULD be taken to improve persistence and long-term preservation of web resources included in an RO-Crate as they can be more difficult to archive or move along with the *RO-Crate root*, and may change intentionally or unintentionally leaving the RO-Crate with incomplete or outdated information.

File Data Entries with an `@id` URI outside the *RO-Crate Root* SHOULD at the time of RO-Crate creation be directly downloadable by a simple retrieval (e.g. HTTP GET), permitting redirections and HTTP/HTTPS authentication. For instance, in the example above, https://zenodo.org/record/3541888 and https://doi.org/10.5281/zenodo.3541888 cannot be used as `@id` above as retrieving these URLs give a HTML landing page rather than the desired PDF as indicated by `encodingFormat`.

As files on the web may change, the timestamp property sdDatePublished SHOULD be included to indicate when the absolute URL was accessed, and derived metadata like encodingFormat and contentSize were considered to be representative:

```
  {
    "@id": "https://zenodo.org/record/3541888/files/ro-crate-1.0.0.pdf",
    "@type": "File",
    "name": "RO-Crate specification",
    "contentSize": "310691",
    "encodingFormat": "application/pdf",
    "sdDatePublished": "2020-04-09T13:09:21+01:00Z"
  }
```

7.3.1 Embedded data entities that are also on the web

File Data Entities may already have a corresponding web presence, for instance a landing page that describes the file, including persistent identifiers (e.g. DOI) resolving to an intermediate HTML page instead of the downloadable file directly.

These can be included for File Data Entities as additional metadata, regardless of whether the File is included in the *RO-Crate Root* directory or exists on the Web, by using the properties:

- identifier for formal identifier strings such as DOIs
- url with a string URL corresponding to a *download* link (if not available, a download landing page) for this file
- subjectOf to a CreativeWork (or WebPage) that mentions this file or its content (but also other resources)
- mainEntityOfPage to a CreativeWork (or WebPage) that primarily describes this file (or its content)

```
{
  "@id": "survey-responses-2019.csv",
  "@type": "File",
  "name": "Survey responses",
  "encodingFormat": "text/csv",
  "url": "http://example.com/downloads/2019/survey-responses-2019.csv",
  "subjectOf": {"@id": "http://example.com/reports/2019/annual-survey.html"}
},
{
  "@id": "https://zenodo.org/record/3541888/files/ro-crate-1.0.0.pdf",
  "@type": "File",
  "name": "RO-Crate specification",
  "encodingFormat": "application/pdf",
  "identifier": "https://doi.org/10.5281/zenodo.3541888",
  "url": "https://zenodo.org/record/3541888"
}
```

7.3.2 Directories on the web; dataset distributions

A *Directory File Entry* or Dataset identifier expressed as an absolute URL on the web can be harder to download than a File because it consists of multiple resources. It is RECOMMENDED that such directories have a complete listing of their content in hasPart, enabling download traversal.

Alternatively, a common mechanism to provide downloads of a reasonably sized directory is as an archive file in formats such as `.zip` or `.tar.gz`, described as a DataDownload.

```
{
  "@id": "lots_of_little_files/",
  "@type": "Dataset",
  "name": "Too many files",
  "description": "This directory contains many small files, that we're not going to desc
  "distribution": {"@id": "http://example.com/downloads/2020/lots_of_little_files.zip"}
```

```
},
{
  "@id": "http://example.com/downloads/2020/lots_of_little_files.zip",
  "@type": "DataDownload",
  "encodingFormat": "application/zip",
  "contentSize": "82818928"
}
```

Similarly, the *RO-Crate root* entity may also provide a distribution URL, in which case the download SHOULD be an archive that contains the *RO-Crate Metadata file*.

In all cases, consumers should be aware that a `DataDownload` is a snapshot that may not reflect the current state of the `Dataset` or RO-Crate.

8 Representing Contextual Entities

The RO-Crate SHOULD contain additional information about *Contextual Entities* for the use of both humans (in `ro-crate-preview.html`) and machines (in `ro-crate-metadata.json`). This also helps to maximize 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.

8.1 Contextual vs Data entities

RO-Crate distinguishes between *Contextual entities* and *Data entities*.

Data entities primarily exist in their own right as a file or directory (which may be in the *RO-Crate Root* directory or downloadable by URL).

Contextual entities however primarily exist outside the digital sphere (e.g. People, Places) or are conceptual descriptions that primarily exists as metadata, like GeoCoordinates and ContactPoint.

Some contextual entities can also be considered data entities - for instance the license property refers to a CreativeWork that can reasonably be downloaded, however a license document is not usually considered as part of research outputs and would therefore typically not be included in hasPart on the root data entity.

Likewise, some data entities may also be described as contextual entities, for instance a `File` that is also a ScholarlyArticle. In such cases the *Contextual Data Entity* MUST be described as a single JSON object in the RO-Crate Metadata JSON `@graph` and SHOULD list both relevant data and contextual types in a `@type` array.

The RO-Crate Metadata JSON `@graph` MUST NOT list multiple entities with the same `@id`; behaviour of consumers of an RO-Crate encountering multiple entities with the same `@id` is undefined.

8.2 Identifiers for contextual entities

A challenge can be how to assign identifiers for contextual entities, that is deciding on their `@id` value.

RO-Crate recommend that if an existing permalink (e.g. `https://orcid.org/0000-0002-1825-0097`) or other absolute URI (e.g. `https://en.wikipedia.org/wiki/Josiah_S._Carberry`) is reasonably unique for that entity, that URI should be used as identifier for the contextual entity in preference of an identifier local to the RO-Crate (e.g. `#josiah` or `#0fa587c6-4580-4ece-a5df-69af3c5590e3`).

Care should be taken to not describe two conceptually different contextual entities with the same identifier - e.g. if `https://en.wikipedia.org/wiki/Josiah_S._Carberry` is a Person it should not also be a CreativeWork (although this example is a fictional person!).

Where a related URL exist that may not be unique enough to serve as identifier, it can instead be added to a contextual entity using the property url.

See the appendix on JSON-LD identifiers for details.

8.3 People

A core principle of Linked data is to use URIs to identify important entities such as people. The following is the minimum recommended way of representing a author of a RO-Crate. The author property MAY also be applied to a directory (Dataset), a File or other CreativeWork entities.

```
{
    "@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, represented as a *Contextual Entity* of type Person.

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

8.4 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"
}
```

8.5 Contact information

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"
}
```

8.6 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:

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

The publication SHOULD be described further as an additional contextual entity of type ScholarlyArticle or CreativeWork.

```
{
  "@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"
}
```

citation MAY also be used with other data and contextual entities:

```
{
  "@id": "communities-2018.csv",
  "@type": "File",
  "name": "Snapshot of RO Community efforts",
  "citation": {"@id": "https://doi.org/10.5281/zenodo.1313066"},
  "encodingFormat": "text/csv"
}
```

A data entity MAY provide a published DOI identifier that, compared with any related publication in citation, primarily captures that file or dataset:

```
{
  "@id": "figure.png",
  "@type": ["File", "ImageObject"],
  "name": "XXL-CT-scan of an XXL Tyrannosaurus rex skull",
  "identifier": "https://doi.org/10.5281/zenodo.3479743",
  "citation": {"@id": "http://ndt.net/?id=19249"},
  "encodingFormat": "image/png"
}
```

8.7 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"
},
```

8.8 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.

Tip

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"
},
```

8.9 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"
}
```

8.9.1 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 Data 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.json",
  "identifier": "ro-crate-metadata.json",
  "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.

In the above (abridged) example, there is no explicit license on the *RO-Crate Metadata File Description*, so the *Root Data Entity* license [GPL 3.0] would apply to RO-Crate JSON-LD statements, except for the statements on the imported http://sws.geonames.org/8152662/, which metadata is re-distributed under license https://creativecommons.org/licenses/by/4.0/.

In this example the CC-BY license requires retaining "a notice that refers to this Public License" and "identification of the creator(s) of the Licensed Material", here respected using `sdLicense` and `sdPublisher`.

As the RO-Crate uses *flattened* JSON-LD, `sdLicense` should be expressed directly on each data/contextual entities where required.

Tip: If metadata is imported from a source licensed as [CC0 Public Domain Dedication][CC0], no `sdLicense` statement is required.

–>

```
Extra metadata such as Exif
---------------------------

Schema.org has a generic extension mechanism for encoding 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)](http://schema.org/ImageObject) of including [Exif](https://en.wikipedia.org/wiki/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.

``` {.json}
{
      "@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
      "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               "
    },
```

8.10 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 its `.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 contextual entity 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"
  },
```

8.11 Subjects & keywords

Subject properties (equivalent to a Dublin Core Subject) on the root data entity 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" },
}
```

8.12 Time

To describe the *time period* which a RO-Crate Data Entity (or the root data entity) 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"
}
```

8.13 Thumbnails

A File or any other entity 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",
    "ImageObject"
  ],
  "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 co
  ],
  "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"
  ],
  "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"
}
```

9 Detailing provenance of entities

9.1 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
  "identifier": "https://confluence.csiro.au/display/ASL/Hovermap",
  "name": "Bentwing"
}
```

Uses CreateAction and UpdateAction type 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"
}
```

9.2 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 (or other data entities) it created as an instrument of 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 as "instrument". 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.
    "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"
    }
  },
```

Tip

If representing command lines, double escape \\ so that JSON preserves the \ character.

If multiple SoftwareApplications have been used in composition, such as from a script or workflow, then the `CreateAction`'s instrument SHOULD rather reference a SoftwareSourceCode which can be further described as explained in the Workflows and scripts section.

9.3 Recording changes to RO-Crates

To record an action which changes an entity's metadata, or changes its state in a publication or other workflow, a CreateAction or UpdateAction SHOULD be associated with a Data Entity or, for the RO-Crate itself, with the root data entity.

A curation Action MUST have at least one object which associates it with either the root data entity `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"
}
```

9.4 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 RepositoryObject.

NOTE

The terms `RepositoryObject` and `RepositoryCollection` are renamed in RO-Crate to avoid collision between other vocabularies and the PCDM terms `Collection` and `Object`. The term `RepositoryFile` is renamed to avoid clash with RO-Crate's `File` mapping to http://schema.org/MediaObject.

WARNING

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
    ],
    "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"
}
```

10 Workflows and Scripts

Scientific workflows and scripts that were used (or can be used) to analyze or generate files contained in an RO-Crate MAY be embedded in an RO-Crate. See also the Provenance section on Software Used to Create Files.

Workflows and *scripts* SHOULD be described using 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 deeper and should therefore be included as `SoftwareSourceCode` in the RO-Crate dataset.

10.1 Describing scripts and workflows

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

- `@type` is an array with at least `File` and `SoftwareSourceCode` as values
- `@id` is a File URI linking to the executable script
- `name`: a human-readable name for the script.

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

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

Short example describing a *script*:

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

Short example describing a *workflow*:

```
{
    "@id": "workflow/retropath.knime",
    "@type": ["File", "SoftwareSourceCode", "ComputationalWorkflow"],
    "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"}
}
```

There is no strong distinction between a *script* and a *workflow*; many computational workflows are written in script-like languages, and many scripts perform a *pipeline* of steps.

Here are some indicators for when a script should be considered a *workflow*:

- It performs a series of steps (*pipeline*)
- The executed steps are mainly external tools or services
- The main work is performed by the steps (script is not algorithmic)
- The steps exchange data in a *dataflow*, typically file inputs/outputs
- The script has well-defined *inputs* and *outputs*, e.g. file arguments

Here are some counter-indicators for when a script might **not** be a workflow:

- The script contains mainly algorithms or logic
- Data is exchanged out of bands, e.g. a SQL database
- The script relies on a particular state of the system (e.g. appends existing files)
- An interactive user interface that controls the actions

10.2 Workflow Runtime and Programming Language

Scripts written in a *programming language*, as well as workflows, generally need a *runtime*; in RO-Crate the runtime SHOULD be indicated using a liberal interpretation of programmingLanguage.

Note that the language and its runtime MAY differ (e.g. different C++ compilers), but for scripts and workflows, frequently the language and runtime are essentially the same, and thus the `programmingLanguage`, implied to be a ComputerLanguage, can also be described as an executable SoftwareApplication:

```
{
  "@id": "scripts/analyse_csv.py",
  "@type": ["File", "SoftwareSourceCode"],
  "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 ComputerLanguage and/or SoftwareApplication 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 `ComputationalWorkflow` or `Script`, by using hasPart to relate additional `SoftwareApplication` or nested `SoftwareSourceCode` contextual entities:

```
{
    "@id": "workflow/analyze.cwl",
    "@type": ["File", "SoftwareSourceCode", "ComputationalWorkflow"],
    "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/"},
    ]
}
```

10.3 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 from the `SoftwareSourceCode` data entity by using `image`, pointing to an ImageObject data entity which is about the `SoftwareSourceCode`:

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

A workflow diagram 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 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"],
  "encodingFormat":  ["image/svg+xml"],
  "name": "Diagram of an ad hoc workflow",
  "about": {"@id": "./"}
}
```

10.4 Complying with Bioschemas Computational Workflow profile

Data entities representing *workflows* (`@type: ComputationalWorkflow`) SHOULD comply with the Bioschemas ComputationalWorkflow profile, where possible.

When complying with this profile, the workflow data entities MUST describe these properties and their related contextual entities: name, programmingLanguage, creator, dateCreated, license, sdPublisher, url, version.

The ComputationalWorkflow profile explains the above and list additional properties that a compliant ComputationalWorkflow data entity SHOULD include: citation, contributor, creativeWorkStatus, description, funding, hasPart, isBasedOn, keywords, maintainer, producer, publisher, runtimePlatform, softwareRequirements, targetProduct

A data entity conforming to the ComputationalWorkflow profile SHOULD declare the versioned profile URI using conformsTo:

```
{ "@id": "workflow/alignment.knime",
  "@type": ["File", "SoftwareSourceCode", "ComputationalWorkflow"],
  "conformsTo":
    {"@id": "https://bioschemas.org/profiles/ComputationalWorkflow/0.5-DRAFT-2020_07_21/"}
  "..": ""
}
```

10.4.1 Describing inputs and outputs

The input and output *parameters* for a workflow or script can be given with `input` and `output` to FormalParameter contextual entities. Note that this entity usually represent a *potential* input/output value in a reusable workflow, much like function parameter definitions in general programming.

If complying with the Bioschemas FormalParameter profile, the *contextual entities* for FormalParameter, referenced by `input` or `output`, MUST describe: name, additionalType, encodingFormat

The Bioschemas FormalParameter profile explains the above and lists additional properties that can be used, including description, valueRequired, defaultValue and identifier.

A contextual entity conforming to the FormalParameter profile SHOULD declare the versioned profile URI using conformsTo, e.g.:

```
{
  "@id": "#36aadbd4-4a2d-4e33-83b4-0cbf6a6a8c5b",
  "@type": "FormalParameter",
  "conformsTo":
    {"@id": "https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-2020_07_21/"},
  "..": ""
}
```

NOTE

`input`, `output` and `FormalParameter` are at time of writing proposed by Bioschemas and not yet integrated in Schema.org

10.5 Complete Workflow Example

The below is an example of an RO-Crate complying with the Bioschemas ComputationalWorkflow profile 0.5:

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
    {
      "@type": "CreativeWork",
      "@id": "ro-crate-metadata.json",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
      "about": {"@id": "./"}
    },
    {
      "@id": "./",
      "@type": "Dataset",
      "hasPart": [
          { "@id": "workflow/retropath.knime" }
      ]
    },
    {
      "@id": "workflow/alignment.knime",
      "@type": ["File", "SoftwareSourceCode", "ComputationalWorkflow"],
      "conformsTo":
        {"@id": "https://bioschemas.org/profiles/ComputationalWorkflow/0.5-DRAFT-2020_07_2
      "name": "Sequence alignment workflow",
      "programmingLanguage": {"@id": "#knime"},
      "creator": {"@id": "#alice"},
      "dateCreated": "2020-05-23",
      "license": { "@id": "https://spdx.org/licenses/CC-BY-NC-SA-4.0"},
      "input": [
        { "@id": "#36aadbd4-4a2d-4e33-83b4-0cbf6a6a8c5b"}
      ],
      "output": [
        { "@id": "#6c703fee-6af7-4fdb-a57d-9e8bc4486044"},
        { "@id": "#2f32b861-e43c-401f-8c42-04fd84273bdf"}
      ],
      "sdPublisher": {"@id": "#workflow-hub"},
```

```
  "url": "http://example.com/workflows/alignment",
  "version": "0.5.0"
},
{
  "@id": "#36aadbd4-4a2d-4e33-83b4-0cbf6a6a8c5b",
  "@type": "FormalParameter",
  "conformsTo": {"@id": "https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-202
  "name": "genome_sequence",
  "valueRequired": true,
  "additionalType": {"@id": "http://edamontology.org/data_2977"},
  "format": {"@id": "http://edamontology.org/format_1929"}
},
{
  "@id": "#6c703fee-6af7-4fdb-a57d-9e8bc4486044",
  "@type": "FormalParameter",
  "conformsTo": {"@id": "https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-202
  "name": "cleaned_sequence",
  "additionalType": {"@id": "http://edamontology.org/data_2977"},
  "encodingFormat": {"@id": "http://edamontology.org/format_2572"}
},
{
  "@id": "#2f32b861-e43c-401f-8c42-04fd84273bdf",
  "@type": "FormalParameter",
  "conformsTo": {"@id": "https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-202
  "name": "sequence_alignment",
  "additionalType": {"@id": "http://edamontology.org/data_1383"},
  "encodingFormat": {"@id": "http://edamontology.org/format_1982"}
},
{
  "@id": "https://spdx.org/licenses/CC-BY-NC-SA-4.0",
  "@type": "CreativeWork",
  "name": "Creative Commons Attribution Non Commercial Share Alike 4.0 International",
  "alternateName": "CC-BY-NC-SA-4.0"
},
{
  "@id": "#knime",
  "@type": "ProgrammingLanguage",
  "name": "KNIME Analytics Platform",
  "alternateName": "KNIME",
  "url": "https://www.knime.com/whats-new-in-knime-41",
  "version": "4.1.3"
},
{
  "@id": "#alice",
  "@type": "Person",
  "name": "Alice Brown"
},
{
  "@id": "#workflow-hub",
  "@type": "Organization",
```

```
      "name": "Example Workflow Hub",
      "url":"http://example.com/workflows/"
    },
    {
      "@id": "http://edamontology.org/format_1929",
      "@type": "Thing",
      "name": "FASTA sequence format"
    },
    {
      "@id": "http://edamontology.org/format_1982",
      "@type": "Thing",
      "name": "ClustalW alignment format"
    },
    {
      "@id": "http://edamontology.org/format_2572",
      "@type": "Thing",
      "name": "BAM format"
    },
    {
      "@id": "http://edamontology.org/data_2977",
      "@type": "Thing",
      "name": "Nucleic acid sequence"
    },
    {
      "@id": "http://edamontology.org/data_1383",
      "@type": "Thing",
      "name": "Nucleic acid sequence alignment"
    }
  ]
}
```

10.6 Appendixes

11 APPENDIX: Changelog

- RO-Crate 1.1.0 https://w3id.org/ro/crate/1.1
  - **Note**: The RO-Crate metadata file is renamed to `ro-crate-metadata.json` to facilitate use of JSON editors. #82 #84
  - Data entities can reference external resources with absolute URI #74
  - Added section on considerations for Web-based Data Entities #74
  - The root dataset is no longer required to be `./` #74
  - RO-Crate Root directory no longer requires payload files #74
  - Workflows and scripts section now aligned with BioSchemas ComputationalWorkflow profile #81 #100
  - Added section Programming with JSON-LD and note that `@type` might be an array #85
  - Added new section Handling relative URI references #73
  - JSON-LD context no longer sets `@base: null` #73
  - Added note on Encoding file paths #77 #80

  - Added section Choosing URLs for ad hoc terms #71 #90
  - Section RO-Crate JSON-LD Media type expanded to suggest HTTP server configuration
  - Update JSON-LD context to Schema.org 10.0
  - Remove HTML use of `relatedLink` property in RepositoryCollection example #91
  - Distinguish between contextual/data entities #94
  - RO-Crate preview HTML no longer needs to "contain same information as JSON-LD" #108
  - Change theme to `jekyll-rtd-theme` and split into multiple pages #95
  - Fixed typos in JSON and English
  - Additional metadata standards showed wrong PCDM namespace #112
  - Citation example expanded 12a6754
  - Terminology adds property, type, entity cc10e28
  - In People `author` can also be applied to `CreativeWork` e086b8b
  - Provenance section on Software-used now points to Workflows section (and vice versa) 5d89872 40de6c7
  - In JSON-LD appendix `@id` should not include `../` 74ef6f1
  - Several sections reviewed to improve language and examples #91 #92 #93 #94 #96 #97 #98 #101 #102 #103 #104 #105 #111 #114
- RO-Crate 1.0.1
  - Fix JSON typo in example

- RO-Crate 1.0.0 https://w3id.org/ro/crate/1.0
  - Description of RO-Crate Metadata File now required
    * .. must use `conformsTo` to indicate RO-Crate version
  - Clarified use of RO-Crate JSON-LD Context
  - Linked Data principles added
  - RO-Crate JSON-LD Context updated to use Schema.org 5.0
  - Workflow and Script now typed with `@type` array instead of `additionalType`
  - Simplified tables of direct properties to list of properties
  - Simplified example of `affiliation`
  - Clarified `#identifiers` and `_:identifiers`
  - Removed links to data.research.uts.edu.au examples
  - Added licensing of metadata
  - Expanded on *Equipment used to create files*
  - Simplified Workflow and Script section
  - Added appendix on JSON-LD
  - Added BagIt implementation notes
  - Added Repository-specific identifiers
  - RO-Crate JSON-LD now licensed CC0
  - RO-Crate JSON-LD self-identifies its version
- RO-Crate 0.2.1
  - Added DOI and document metadata
- RO-Crate 0.2.0 https://w3id.org/ro/crate/0.2
  - Based on two earlier specifications:

    - RO Lite 0.1.0
    - DataCrate Specification version 1.0.0 2019-04-12
  - RO-Crate Metadata file has been renamed to `ro-crate-metadata.jsonld` instead of `CATALOG.json` (DataCrate) or `manifest.jsonld` (RO-Lite)
  - RO Crate Website renamed to `ro-crate-metadata.html` instead of DataCrate's `CATALOG.html`
  - "RO-Lite" and "DataCrate" renamed to "RO-Crate"
  - Multiple examples and clarifications added
  - RO-Crate directory no longer requires BagIt structure
  - Added section on Workflows and scripts
  - RO-Crate Metadata File must describe itself as being `about` the RO-Crate Dataset.
  - JSON-LD should now be flattened and then compacted (RO-Lite allowed any JSON-LD, DataCrate required flattened)

12 APPENDIX: Implementation notes

12.1 Programming with JSON-LD

When implementing tools to work with RO-Crate it is not necessary to use JSON-LD software libraries, however, programmers should keep in mind the following:

- ***RO-Crate JSON-lD* has a flat structure**; every entity is a JSON object directly within the `@graph` array in the *RO-Crate Metadata File*. A useful strategy when processing a crate is to build a look-up table and/or function so that entities can be found via their ID, for example provide a method such as `getEntity(id)` which returns an entity by its id or a `null` value if it's not there.
- **Code defensively**. Code should not assume that values will always be a String; values for properties may be single scalar values such as strings or integers (`"2"` or 2), or references to other entities such as `{"@id", "_:1"}` (where the referenced entity may or may not be described in the crate, see the point above about having a `getEntity()` method).
- **Read the *whole* specification**. The RO-Crate specification addresses common use cases individually, introducing aspects of the specification as in a progressive manner. Some key points, such as *entities may have more than one value for `@type`*, may not be apparent from a quick reading.

12.2 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.

12.2.1 Adding RO-Crate to Bagit

RO-Crate can be combined with BagIt simply by placing the RO-Crate files within the BagIt payload (`data/`) directory.

```
<BagIt base 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/                    # Payload: RO-Crate root directory
      |   ro-crate-metadata.json          # 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
```

The Bag declaration `bagit.txt` MUST be present, the main role of this file is to mark the folder as a bag according to RFC8493. The file SHOULD have this fixed content in UTF-8:

```
BagIt-version: 1.0
Tag-File-Character-Encoding: UTF-8
```

The *BagIt base directory* containing `bagit.txt` can have any name, and can be archived/transferred in any way, e.g. within a ZIP archive, SFTP or even be exposed on the web.

The manifest file contains file checksums; the BagIt specifications recommends SHA-512 as default algorithm, that is `manifest-sha512.txt` SHOULD be present.

The BagIt manifest file MUST list the checksum of *all* payload files in `data/` and its subdirectories. Where `data/` is also the RO-Crate Root the manifest therefore MUST include `ro-crate-metadata.json`:

```
41846747…ee71  data/ro-crate-metadata.json
e1105ed0…5e13  data/chipseq_20200910.json
37fd3a02…bb95  data/results/pipeline_info/design_reads.csv
```

NOTE

The SHA-512 checksums have been shortened in the above example.

Creating the manifest file without using BagIt tools/libraries can be done using the equivalent of:

```
$ find data -type f -print0 | xargs -0 sha512sum > manifest-sha512.txt
```

Similarly checking the payload directory:

```
$ sha512sum --quiet -c manifest-sha512.txt
data/chipseq_20200910.json: FAILED
data/ro-crate-metadata.json: FAILED
sha512sum: WARNING: 2 computed checksums did NOT match
```

The BagIt manifest complements the RO-Crate structure as it provide a complete listing of all payload files with cryptographically strong checksums, ensuring the crate has been fully archived/transferred, which the weak CRC-32 checksum (TCP/IP, ZIP, gzip) is insufficient to guarantee, particularly for large crates.

To ensure the manifest file itself is complete, it is RECOMMENDED to include its checksum in `tagmanifest-sha512.txt`:

```
b0556450…8802  bag-info.txt
000b27e3…c52e  manifest-sha512.txt
```

Warning

The BagIt manifest is intended to detect "bit rot" and accidental damage, it does not provide proof the RO-Crate has not been deliberately tampered with, as a malicious actor can also update the checksums.

Guarding against such scenarious would require additional cryptographic measures, e.g.
`gpg --detach-sign --armor --output tagmanifest-sha512.txt.asc tagmanifest-sha512.txt` in combination with a secure PGP key exchange or equivalent trust network.

12.2.1.1 Base URI in BagIt

The arcp specification suggests how BagIt UUID identifiers can be used to calculate the base URI of a bag, see section Establishing a base URI inside a ZIP file. For this purpose it is RECOMMENDED that `bag-info.txt` includes a fresh UUID like:

```
External-Identifier: urn:uuid:24e51ca2-5067-4598-935a-dac4e327d05a
```

12.2.1.2 Referencing external files

The BagIt fetch file MAY be used to reference files to be downloaded into particular `data/` paths to *complete* the bag. These files may be large, require authentication or otherwise inconvenient to transfer within the BagIt folder.

Example `fetch.txt` using Git LFS:

```
https://media.githubusercontent.com/…/SPT5_INPUT_R1.bigWig 963489 data/results/SPT5_INPUT_
```

BagIt tools can help complete the bag and verify the checksum of the downloaded files according to the manifest.

The RO-Crate contained in `data/` MAY describe the bag with data entities as if the bag was *complete*, even if the large file is not (yet) present:

```
{
  "@id": "results/SPT5_INPUT_R1.bigWig",
  "@type": "File",
  "name": "Normalized SPT5_INPUT_R1 bigWig for genome browsers",
```

```
    "encodingFormat": {"@id": "http://edamontology.org/format_3006"},
    "url": "https://media.githubusercontent.com/media/biocompute-objects/bco-ro-example-ch
  }
```

It is RECOMMENDED that the `url` is provided in the data entity and consistent with the line in `fetch.txt` in case the RO-Crate is transferred outside its BagIt container.

The `fetch.txt` approach can also be useful where other files in the RO-Crate reference a downloadable file by relative paths within `data/`, even if this file is not itself described in the RO-Crate metadata.

12.2.1.3 Snapshots of external files

As an alternative to the above, web-based data entities can be used in the RO-Crate:

```
  {
    "@id": "https://media.githubusercontent.com/media/biocompute-objects/bco-ro-example-ch
    "@type": "File",
    "name": "Normalized SPT5_INPUT_R1 bigWig for genome browsers",
    "encodingFormat": {"@id": "http://edamontology.org/format_3006"}
  }
```

The above data entity MAY be combined with `fetch.txt` in the BagIt base directory:

```
https://media.githubusercontent.com/…/SPT5_INPUT_R1.bigWig 963489 data/snapshots/SPT5_INPU
```

In this case the file `data/snapshots/SPT5_INPUT_R1.bigWig` may be present, but unknown by RO-Crate; BagIt contains a checksummed snapshot of the web resource. Compared with the first approach, the RO-Crate is here primarily pointing at a web resource which is allowed to change without causing a BagIt checksum error.

12.2.2 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.json    # 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
  |   bag1/                     # "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` in this scenario would have an `@id` of `bag1/data/example.txt`:

```
{
  "@id": "bag1/data/example.txt",
  "name": "Example file"
}
```

12.3 Repository-specific identifiers

Root Data Entities MAY include 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
}

 {
   "@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"
 }
```

13 APPENDIX: RO-Crate JSON-LD

It is not necessary to use JSON-LD tooling to generate or parse the *RO-Crate Metadata File*, although JSON-LD tools may make it easier to conform to this specification, e.g. handling relative URIs. It is however 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 example below shows the overall structure of a flattened, compacted *RO-Crate Metadata File* where `@context` refers to the *RO-Crate JSON-LD Context*, while `@graph` is a flat array of the entities that constitute this RO-Crate.

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [

    {
```

```
    "@type": "CreativeWork",
    "@id": "ro-crate-metadata.json",
    "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
    "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 the individual sections for data entities and contextual entities.

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

13.1 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 URI* like http://example.com/, or a *URI path* relative to the RO-Crate root directory. Although legal in JSON-LD, `@id` paths in RO-Crate SHOULD NOT use `../` to climb out of the *RO-Crate Root*, rather such references SHOULD be translated to absolute URIs. See also section Core Metadata for Data Entities.

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 JSON objects as direct children of the `@graph` element rather than being nested under another object or array. Properties referencing entities must use a JSON object with `@id` as the only key, e.g. `"author": {"@id": "https://orcid.org/0000-0002-1825-0097"}`

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"}`.

13.2 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/1.1/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/1.1/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/1.1/context",
  "@graph": [
    {
      "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "description": "RO-Crate Metadata File Descriptor (this file)",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
      "about": {"@id": "./"}
    }
  ]
}
```

The above is equivalent to the following 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.json",
      "@type": "CreativeWork",
      "description": "RO-Crate Metadata File Descriptor (this file)",
      "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
      "about": {"@id": "./"}
    }
  ]
}
```

Note that `conformsTo` is retained to indicate which version of RO-Crate specification the root data entity conforms to.

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 download the JSON-LD contexts corresponding to each version of this specification.

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* (see also notes on handling relative URI references below).

Tip

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

13.3 RO-Crate JSON-LD Media type

The media type `application/ld+json` for `ro-crate-metadata.json` 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.json HTTP/1.1


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

Note that most web servers will however serve `*.json` as `Content-Type: application/json`.

Requesting the RO-Crate metadata file from a browser may also need permission through CORS header `Access-Control-Allow-Origin` (however extra care should be taken if the RO-Crates require access control).

To change the configuration of **Apache HTTPD 2**, add the following to `.htaccess` or equivalent config file:

```
<Files "ro-crate-metadata.json">
  ForceType 'application/ld+json;profile="http://www.w3.org/ns/json-ld#flattened http://ww

  Header set Access-Control-Allow-Origin *
  Header set Access-Control-Expose-Headers "Content-Length,Content-Range,Content-Type"
</Files>
```

For **NGINX**, try:

```
location ~ ro-crate-metadata.json$ {
        types { } default_type 'application/ld+json;profile="http://www.w3.org/ns/json-ld#

        add_header 'Access-Control-Allow-Origin' '*';
        add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range,Content-T
}
```

For **Content-Delivery Networks** (e.g. GitHub pages) a symbolic link to `ro-crate-metadata.jsonld` may help to create an alias that can be served as `application/ld+json`:

```
ln -s ro-crate-metadata.json ro-crate-metadata.jsonld
```

13.4 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 types) 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. When 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 intends 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. To read more about best practices for content negotiation of vocabularies, we refer the reader to Best Practice Recipes for Publishing RDF Vocabularies.

```
{
  "@context": [
    "https://w3id.org/ro/crate/1.1/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 a page describing an XML element or an attribute from an XML schema, pending the publication of a formal ontology.

13.5 Adding new or ad hoc vocabulary terms

Context terms must ultimately map to HTTP(s) URIs which poses challenges for crate-authors wishing to use their own vocabularies.

RO-Crate provides some strategies to add a new term (a Class or Property) that is not in Schema.org or another published vocabulary, so that there is a stable URI that can be added to the @context.

13.5.1 Choosing URLs for ad hoc terms

For projects that have their own web-presence, URLs MAY be defined there and SHOULD resolve to useful content. For example for a project with web page

https://criminalcharacters.com/ the property `education` could have a URL: https://criminalcharacters.com/vocab#education which resolves to an HTML page that explains the term using HTML anchors:

```
<div id="education">
  <h1>Property: education</h1>
  <p>Literacy of prisoner. Prison authorities would record the prisoner's statement as to
  </p>
</div>
```

Tip

Ensure you have a consistent use of `http` or `https` (preferring https) as well as consistent path `/vocab` vs `/vocab/` vs `/vocab/index.html` (preferring the shortest that is also visible in browser).

For ad hoc terms where the crate author does not have the resources to create and maintain an HTML page, authors may use the RO-Crate public namespace (`https://w3id.org/ro/terms/`) to reserve their terms. For example, an ad-hoc URL MAY be used in the form `https://w3id.org/ro/terms/criminalcharacters#education` where `criminalcharacters` is acting as a *namespace* for one or more related terms like `education`. Ad-hoc namespaces under `https://w3id.org/ro/terms/` are available on first-come-first-serve basis; to avoid clashes, namespaces SHOULD be registered by submitting terms and definitions to the RO-Crate terms project.

In both cases, to use an ad-hoc term in an RO-Crate, the URI MUST be included in the local context:

```
{
  "@context": [
    "https://w3id.org/ro/crate/1.1/context",
    {"education": "https://criminalcharacters.com/vocab#education",
     "interests": "https://w3id.org/ro/terms/criminalcharacters#interests"},
  ],
  "@graph": [ ... ]
}
```

13.5.2 Add local definitions of ad hoc terms

Following the conventions used by Schema.org, ad-hoc terms SHOULD also include definitions in the RO-Crate with at minimum:

- `@type` of either `Class` (contextual entity type) or `Property` (attribute of an contextual entity)
- `rdfs:label` with the human readable version of the term, e.g. `makesFood` has label `makes food`
- `rdf:comment` documenting and clarifying the meaning of the term. For instance the term `sentence` in a prisoner vocabulary will have a different explanation than `sentence` in a linguistic vocabulary.

```
{
    "@id": "https://criminalcharacters.com/vocab#education",
```

```
    "@type": "rdf:Property",
    "rdfs:label": "education",
    "rdf:comment": "Literacy of prisoner. ..."
}
```

Tip

It is **not** a requirement to use English for the terms, labels or comments.

More information about the relationship of this term to other terms MAY be provided using domainIncludes, rangeIncludes, rdfs:subClassOf following the conventions used in the Schema.org schema.

14 APPENDIX: Handling relative URI references

The *RO-Crate Metadata File* use *relative URI references* to identify files and directories contained within the *RO-Crate Root* and its children. As described in section Describing entities in JSON-LD above, relative URI references are also frequently used for identifying *Contextual entities*.

When using JSON-LD tooling and RDF libraries to consume or generate RO-Crates, extra care should be taken to ensure these URI references are handled correctly.

For this, a couple of scenarios are sketched below with recommendations for consistent handling:

14.1 Flattening JSON-LD from nested JSON

If performing JSON-LD flattening to generate a valid *RO-Crate Metadata File*, add `@base: null` to the input JSON-LD `@context` array to avoid expanding relative URI references. The flattening `@context` SHOULD NOT need `@base: null`.

Example, this JSON-LD is in compacted form which may be beneficial for processing, but is not yet valid *RO-Crate Metadata File* as it has not been flattened into a `@graph` array.

```
{
  "@context": [
    {"@base": null},
    "https://w3id.org/ro/crate/1.1/context"
  ],
  "@id": "ro-crate-metadata.json",
  "@type": "CreativeWork",
  "description": "RO-Crate Metadata File Descriptor (this file)",
  "conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"},
  "about": {
    "@id": "./",
    "@type": "Dataset",
    "name": "Example RO-Crate",
```

```
    "description": "The RO-Crate Root Data Entity",
    "hasPart": [
      { "@id": "data1.txt",
        "@type": "File",
        "description": "One of hopefully many Data Entities",
      },
      { "@id": "subfolder/",
        "@type": "Dataset"
      }
    ]
  }
}
```

Performing JSON-LD flattening with:

```
{ "@context":
     "https://w3id.org/ro/crate/1.1/context"
}
```

Results in a valid *RO-Crate JSON-LD* (actual order in `@graph` may differ):

```
{
  "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
    {
      "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "conformsTo": {
        "@id": "https://w3id.org/ro/crate/1.1"
      },
      "about": {
        "@id": "./"
      },
      "description": "RO-Crate Metadata File Descriptor (this file)"
    },
    {
      "@id": "./",
      "@type": "Dataset",
      "description": "The RO-Crate Root Data Entity",
      "hasPart": [
        {
          "@id": "data1.txt"
        },
        {
          "@id": "subfolder/"
        }
      ],
      "name": "Example RO-Crate"
    },
    {
      "@id": "data1.txt",
      "@type": "File",
```

```
      "description": "One of hopefully many Data Entities"
    },
    {
      "@id": "subfolder/",
      "@type": "Dataset"
    }
  ]
}
```

NOTE

The saved *RO-Crate JSON-LD* SHOULD NOT include `{@base: null}` in its `@context`.

14.2 Expanding/parsing JSON-LD keeping relative referencing

JSON-LD Expansion can be used to resolve terms from the `@context` to absolute URIs, e.g. `http://schema.org/description`. This may be needed to parse extended properties or for combinations with other Linked Data.

This algorithm would normally also expand `@id` fields based on the current base URI of the *RO-Crate Metadata File*, but this may be a temporary location like `file:///tmp/rocrate54/ro-crate-metadata.json`, meaning `@id: subfolder/` becomes `file:///tmp/rocrate54/subfolder/` after JSON-LD expansion.

To avoid absoluting local identifiers, before expanding, augment the JSON-LD `@context` to ensure it is an array that includes `{"@base": null}`.

For example, expanding this JSON-LD:

```
{
  "@context": [
    "https://w3id.org/ro/crate/1.1/context",
    {"@base": null}
  ],
  "@graph": [
    {
      "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "conformsTo": {
        "@id": "https://w3id.org/ro/crate/1.1"
      },
      "about": {
        "@id": "./"
      },
      "description": "RO-Crate Metadata File Descriptor (this file)"
    },
    {
      "@id": "./",
      "@type": "Dataset",
      "description": "The RO-Crate Root Data Entity",
```

```
      "hasPart": [
        {
          "@id": "data1.txt"
        },
        {
          "@id": "subfolder/"
        }
      ],
      "name": "Example RO-Crate"
    }
  ]
}
```

Results in a expanded form without `@context`, using absolute URIs for properties and types, but retains relative URI references for entities within the *RO-Crate Root*:

```
[
  {
    "@id": "ro-crate-metadata.json",
    "@type": [
      "http://schema.org/CreativeWork"
    ],
    "http://schema.org/about": [
      {
        "@id": "./"
      }
    ],
    "http://purl.org/dc/terms/conformsTo": [
      {
        "@id": "https://w3id.org/ro/crate/1.1"
      }
    ],
    "http://schema.org/description": [
      {
        "@value": "RO-Crate Metadata File Descriptor (this file)"
      }
    ]
  },
  {
    "@id": "./",
    "@type": [
      "http://schema.org/Dataset"
    ],
    "http://schema.org/description": [
      {
        "@value": "The RO-Crate Root Data Entity"
      }
    ],
    "http://schema.org/hasPart": [
      {
```

```
        "@id": "data1.txt"
      },
      {
        "@id": "subfolder/"
      }
    ],
    "http://schema.org/name": [
      {
        "@value": "Example RO-Crate"
      }
    ]
  }
]
```

Note

`@base: null` will not relativize existing absolute URIs that happen to be contained by the *RO-Crate Root* (see section Relativizing absolute URIs within RO-Crate Root).

Tip

Most RDF parsers supporting JSON-LD will perform this kind of expansion before generating triples, but not all RDF stores or serializations support relative URI references. Consider using an alternative `@base` as detailed in sections below.

14.3 Establishing absolute URI for RO-Crate Root

When loading *RO-Crate JSON-LD* as RDF, or combining the crate's Linked Data into a larger JSON-LD, it is important to ensure correct base URI to resolve URI references that are relative to the *RO-Crate Root*.

Note

When retrieving an RO-Crate over the web, servers might have performed HTTP redirections so that the base URI is different from what was requested. It is RECOMMENDED to follow section Establishing a Base URI of RFC3986 before resolving relative links from the *RO-Crate Metadata File*.

For instance, consider this HTTP redirection from a permalink (simplified):

```
GET https://w3id.org/ro/crate/1.0/crate HTTP/1.1

HTTP/1.1 301 Moved Permanently
Location: https://www.researchobject.org/ro-crate/1.0/ro-crate-metadata.jsonld

GET https://www.researchobject.org/ro-crate/1.0/ro-crate-metadata.jsonld HTTP/1.1

HTTP/1.1 200 OK
Content-Type: application/ld+json

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

```
    "@graph": [
      {
        "@id": "ro-crate-metadata.jsonld",
        "@type": "CreativeWork",
        "conformsTo": {
          "@id": "https://w3id.org/ro/crate/1.0"
        },
        "about": {
          "@id": "./"
        },
        "license": {
          "@id": "https://creativecommons.org/publicdomain/zero/1.0/"
        }
      },
      {
        "@id": "./",
        "@type": "Dataset",
        "hasPart": [
          {
            "@id": "index.html"
          }
      }
    ]
}
```

Following redirection we see that:

- *Base URI* of the *RO-Crate Metadata File* becomes `https://www.researchobject.org/ro-crate/1.0/`
- The absolute URI for `index.html` resolves to `https://www.researchobject.org/ro-crate/1.0/index`
  - ..rather than `https://w3id.org/ro/crate/1.0/index.html` which would not redirect correctly

This example also use RO-Crate 1.0, where the *RO-Crate Metadata File* is called `ro-crate-metadata.jsonld` instead of `ro-crate-metadata.json`. Note that the recommended algorithm to find the Root Data Entity is agnostic to the actual filename.

14.4 Finding RO-Crate Root in RDF triple stores

When parsing *RO-Crate JSON-LD* as RDF, where the RDF framework performs resolution to absolute URIs, it may be difficult to find the *RO-Crate Root* in the parsed triples.

The algoritm proposed in section Root Data Entity allows finding the RDF resource describing `ro-crate-metadata.json`, independent of its parsed base URI. We can adopt this for RDF triples, thus finding crates conforming to this specification can be queried with SPARQL:

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX schema:  <http://schema.org/>

SELECT ?crate ?metadatafile
```

```
WHERE {
  ?crate        a                  schema:Dataset .
  ?metadatafile schema:about       ?crate .
  ?metadatafile dcterms:conformsTo <https://w3id.org/ro/crate/1.1> .
}
```

..or (less efficient) for any RO-Crate version:

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX schema:  <http://schema.org/>

SELECT ?crate ?metadatafile ?spec
WHERE {
  ?crate        a                  schema:Dataset .
  ?metadatafile schema:about       ?crate .
  ?metadatafile dcterms:conformsTo ?spec .

  FILTER STRSTARTS(str(?spec), "https://w3id.org/ro/crate/")
}
```

14.5 Parsing as RDF with a different RO-Crate Root

When parsing a *RO-Crate Metadata File* into RDF triples, for instance uploading it to a *graph store* like Apache Jena's Fuseki, it is important to ensure consistent *base URI*:

- Some RDF stores and RDF formats don't support relative URI references in triples (see RDF 1.1 note on IRIs)
- The *RO-Crate Root* may depend on where the *RO-Crate Metadata File* was parsed from, e.g. `<file:///tmp/ro-crate-metadata.json>` (file) or `<http://localhost:3030/test/ro-crate-metadata.json>` (web upload)
- Parsing multiple RO-Crates into the same RDF graph, using same base URI, may merge them into the same RO-Crate
- `ro-crate-metadata.json` may not be recognized as JSON-LD and must be renamed to `ro-crate-metadata.jsonld`
- Web servers hosting `ro-crate-metadata.json` may not send the JSON-LD *Content-Type*
- If base URI is not correct it may be difficult to find the corresponding file and directory paths from an RDF query returning absolute URIs

Tip

If the RDF library can parse the *RO-Crate JSON-LD* directly by retrieving from a `http`/`https` URI of the *RO-Crate Metadata File* it should calculate the correct base URI as detailed in section Establishing absolute URI for RO-Crate Root and you should **not** need to override the base URI as detailed here.

If a web-based URI for the *RO-Crate root* is known, then this can be supplied as a *base URI*. Most RDF tools support a `--base` option or similar. If this is not possible, then the `@context` of the `RO-Crate JSON-LD` can be modified by ensuring the `@context` is an array that sets the desired `@base`:

```
{
  "@context": [
    "https://w3id.org/ro/crate/1.1/context",
    {"@base": "http://example.com/crate255/"}
  ],
  "@graph": [
    {
      "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "conformsTo": {
        "@id": "https://w3id.org/ro/crate/1.1"
      },
      "about": {
        "@id": "./"
      }
    },
    {
      "@id": "./",
      "@type": "Dataset",
      "name": "Example RO-Crate"
    },
    {
      "@id": "data1.txt",
      "@type": "File",
      "description": "One of hopefully many Data Entities"
    },
    {
      "@id": "subfolder/",
      "@type": "Dataset"
    }
  ]
}
```

Parsing this will generate triples like below using `http://example.com/crate255/` as the *RO-Crate Root* (shortened):

```
<http://example.com/crate255/ro-crate-metadata.json>
  <http://purl.org/dc/terms/conformsTo>
  <https://w3id.org/ro/crate/1.1> .

<http://example.com/crate255/ro-crate-metadata.json>
  <http://schema.org/about>
  <http://example.com/crate255/> .

<http://example.com/crate255/>
  <http://schema.org/name>
  "Example RO-Crate" .

<http://example.com/crate255/>
  <http://schema.org/hasPart>
  <http://example.com/crate255/data1.txt> .
```

```
<http://example.com/crate255/>
  <http://schema.org/hasPart>
  <http://example.com/crate255/subfolder/> .

<http://example.com/crate255/data1.txt>
 <http://schema.org/description>
 "One of hopefully many Data Entities" .
```

Generating a *RO-Crate JSON-LD* from such triples can be done by first finding the RO-Crate Root and then use it as base URI to relativize absolute URIs within RO-Crate Root.

14.6 Establishing a base URI inside a ZIP file

An RO-Crate may have been packaged as a ZIP file or similar archive. RO-Crates may exist in a temporary file path which should not determine its identifiers.

When parsing such crates it is recommended to use the Archive and Package (arcp) URI scheme to establish a temporary/location-based UUID or hash-based (SHA256) *base URI*.

For instance, given a randomly generated UUID `b7749d0b-0e47-5fc4-999d-f154abe68065` we can use `arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/` as the `@base`:

```
{
  "@context": [
    "https://w3id.org/ro/crate/1.1/context",
    {"@base": "arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/"}
  ],
  "@graph": [
    {
      "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "conformsTo": {
        "@id": "https://w3id.org/ro/crate/1.1"
      },
      "about": {
        "@id": "./"
      }
    },
    {
      "@id": "./",
      "@type": "Dataset",
      "description": "The RO-Crate Root Data Entity",
      "hasPart": [
        {
          "@id": "data1.txt"
        },
        {
```

```
          "@id": "subfolder/"
        }
      ],
      "name": "Example RO-Crate"
    },
    {
      "@id": "data1.txt",
      "@type": "File",
      "description": "One of hopefully many Data Entities"
    },
    {
      "@id": "subfolder/",
      "@type": "Dataset"
    }
  ]
}
```

Parsing this as RDF will generate triples including:

```
<arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/ro-crate-metadata.json>
  <http://schema.org/about>
  <arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/> .

<arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/>
  <http://schema.org/hasPart>
  <arcp://uuid,b7749d0b-0e47-5fc4-999d-f154abe68065/data1.txt> .
```

Here consumers can assume `/` is the *RO-Crate Root* and generating relative URIs can safely be achieved by search-replace as the arcp URI is unique. Saving *RO-Crate JSON-LD* from the triples can be done by using the arcp URI to relativize absolute URIs within RO-Crate Root.

Tip

Bagit: The arcp specification suggests how BagIt identifiers can be used to calculate the base URI. See also section Combining with other packaging schemes - note that in this approach the *RO-Crate Root* will be the payload folder `/data/` under the calculated arcp base URI.

14.7 Relativizing absolute URIs within RO-Crate Root

Some applications may prefer working with absolute URIs, e.g. in a joint graph store or web-based repository, but should relativize URIs within the *RO-Crate Root* before generating the *RO-Crate Metadata File*.

Assuming a repository at `example.com` has JSON-LD with absolute URIs:

```
{
  "@context": "https://w3id.org/ro/crate/1.1",
  "@graph": [
    {
      "@id": "http://example.com/crate415/ro-crate-metadata.json",
      "@type": "CreativeWork",
```

```
      "conformsTo": {
        "@id": "https://w3id.org/ro/crate/1.1"
      },
      "about": {
        "@id": "http://example.com/crate415/"
      },
    },
    {
      "@id": "http://example.com/crate415/",
      "@type": "Dataset",
      "description": "The RO-Crate Root Data Entity",
      "hasPart": [
        {
          "@id": "http://example.com/crate415/data1.txt"
        },
        {
          "@id": "http://example.com/crate415/subfolder/"
        }
      ],
      "name": "Example RO-Crate"
    }
  ]
}
```

Then performing JSON-LD flattening with this `@context`:

```
{ "@context": [
    {"@base": "http://example.com/crate415/"},
     "https://w3id.org/ro/crate/1.1"
  ]
}
```

Will output *RO-Crate JSON-LD* with relative URIs:

```
{
  "@context": [
    {
      "@base": "http://example.com/crate415/"
    },
    "https://w3id.org/ro/crate/1.1"
  ],
  "@graph": [
    {
      "@id": "./",
      "@type": "Dataset",
      "description": "The RO-Crate Root Data Entity",
      "hasPart": [
        {
          "@id": "data1.txt"
        },
        {
          "@id": "subfolder/"
```

```
        }
      ],
      "name": "Example RO-Crate"
    },
    {
      "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "conformsTo": {
        "@id": "https://w3id.org/ro/crate/1.1"
      },
      "about": {
        "@id": "./"
      }
    }
  ]
}
```

Warning

This method would also relativize URIs outside the *RO-Crate Root* that are on the same host, e.g. `http://example.com/crate255/other.txt` would become `../create255/other.txt` - this can particularly be a challenge with local `file:///` URIs.

15 References

- [BagIt]: https://en.wikipedia.org/wiki/BagIt
- [BagIt profile]: https://github.com/ruebot/bagit-profiles
- [BIBO]: http://purl.org/ontology/bibo/interviewee
- [conformsTo]: http://purl.org/dc/terms/conformsTo
- [CURIE]: https://www.w3.org/TR/curie/
- [DataCite]: https://www.datacite.org/
- [DataCite Schema]: https://schema.datacite.org/
- [DataCite Schema v4.0]: https://schema.datacite.org/meta/kernel-4.0/metadata.xsd
- [DCAT]: https://www.w3.org/TR/vocab-dcat/
- [Exif]: https://en.wikipedia.org/wiki/Exif
- [Flattened Document Form]: https://json-ld.org/spec/latest/json-ld/#flattened-document-form
- [FRAPO]: https://www.sparontologies.net/ontologies/frapo
- [geonames]: https://www.geonames.org/
- [git]: https://git-scm.com/
- [hasFile]: https://pcdm.org/2016/04/18/models#hasFile

- [hasMember]: https://pcdm.org/2016/04/18/models#hasMember
- [isOutputOf]: https://sparontologies.github.io/frapo/current/frapo.html#d4e526
- [JSON]: http://json.org/
- [JSON-LD]: https://json-ld.org/
- [linked data]: https://en.wikipedia.org/wiki/Linked_data
- [OCFL]: https://ocfl.io/
- [OCFL Object]: https://ocfl.io/1.0/spec/#object-spec
- [ORCID]: https://orcid.org
- [Pairtree]: https://confluence.ucop.edu/display/Curation/PairTree
- [Pairtree specification]: https://confluence.ucop.edu/display/Curation/PairTree?preview=/14254128/16973838/PairtreeSpec.pdf
- [PCDM]: https://github.com/duraspace/pcdm/wiki
- [Pronom]: https://www.nationalarchives.gov.uk/PRONOM/Default.aspx
- [RepositoryCollection]: https://pcdm.org/2016/04/18/models#Collection
- [RepositoryObject]: https://pcdm.org/2016/04/18/models#Object
- [ResearchObject]: https://www.researchobject.org/
- [Schema.org]: http://schema.org
- [WorkflowSketch]: http://wf4ever.github.io/ro/2016-01-28/roterms/#Sketch
- [Omeka]: https://omeka.org
- [Linked Data principles]: https://5stardata.info/en/
- [open standards from W3C]: https://www.w3.org/standards/
- [CC-BY]: https://creativecommons.org/licenses/by/4.0/
- [GPL 3.0]: https://www.gnu.org/licenses/gpl-3.0
- [CC0]: https://creativecommons.org/publicdomain/zero/1.0/
- [Exif]: https://en.wikipedia.org/wiki/Exif
- [HTML 5]: https://www.w3.org/TR/html52/
- [W3C RDF 1.1 formats]: https://www.w3.org/TR/rdf11-primer/
- [Research Organization Registry URIs]: https://ror.org/
- [Google Dataset Search]: https://developers.google.com/search/docs/data-types/dataset
- [JSON-LD 1.0]: https://www.w3.org/TR/2014/REC-json-ld-20140116/

- [flattened]: https://www.w3.org/TR/json-ld/#flattened-document-form
- [compacted]: https://www.w3.org/TR/json-ld/#compacted-document-form
- [FORCE11 Data Citation Principles]: https://doi.org/10.25490/a97f-egyk
- [RFC3986 section 3.3]: https://tools.ietf.org/html/rfc3986#section-3.3
- [JSON-LD context]: https://www.w3.org/TR/json-ld/#the-context
- [BagIt UUID identifiers]: https://tools.ietf.org/html/draft-soilandreyes-arcp-03#appendix-A.4
- [domainIncludes]: http://schema.org/domainIncludes
- [rangeIncludes]: http://schema.org/rangeIncludes
- [Schema.org schema]: https://schema.org/version/latest/schemaorg-current-http.jsonld
- [rdfs:subClassOf]: https://www.w3.org/TR/rdf-schema/#ch_subclassof
- [JSON-LD tooling]: https://json-ld.org/#developers
- [JSON-LD flattening]: https://www.w3.org/TR/json-ld-api/#flattening-algorithm
- [JSON-LD Expansion]: https://www.w3.org/TR/json-ld-api/#expansion
- [JSON-LD base URI]: https://www.w3.org/TR/json-ld11/#base-iri
- [JSON-LD expanded form]: https://www.w3.org/TR/json-ld11/#expanded-document-form
- [RFC3986 base URI]: http://tools.ietf.org/html/rfc3986#section-5.1
- [SPARQL]: https://www.w3.org/TR/sparql11-query/
- [RDF triples]: https://www.w3.org/TR/rdf11-concepts/
- [Apache Jena]: https://jena.apache.org/
- [Fuseki]: https://jena.apache.org/documentation/fuseki2/
- [RDF 1.1 note on IRIs]: https://www.w3.org/TR/rdf11-concepts/#note-iris
- [ARCP]: https://tools.ietf.org/id/draft-soilandreyes-arcp-03.html
- [ARCP BagIt]: https://tools.ietf.org/html/draft-soilandreyes-arcp-03#appendix-A.4
- [media type]: https://www.iana.org/assignments/media-types
- [JSON-LD media type]: https://www.w3.org/TR/json-ld/#application-ld-json
- [RFC7231 response]: https://tools.ietf.org/html/rfc7231#section-3.1.1.5
- [BIBO ontology]: http://neologism.ecs.soton.ac.uk/bibo.html

- [ro-terms]: https://github.com/ResearchObject/ro-terms
- [function parameter definitions]: https://en.wikipedia.org/wiki/Parame ter_(computer_programming)
- [Best Practice Recipes for Publishing RDF Vocabularies]: https://www. w3.org/TR/swbp-vocab-pub/
- [Action]: http://schema.org/Action
- [ActionStatusType]: http://schema.org/ActionStatusType
- [ActiveActionStatus]: http://schema.org/ActiveActionStatus
- [Class]: http://schema.org/Class
- [CompletedActionStatus]: http://schema.org/CompletedActionStatus
- [ComputerLanguage]: http://schema.org/ComputerLanguage
- [CreateAction]: http://schema.org/CreateAction
- [CreativeWork]: http://schema.org/CreativeWork
- [DataDownload]: http://schema.org/DataDownload
- [DateTime]: https://schema.org/DateTime
- [Dataset]: http://schema.org/Dataset
- [FailedActionStatus]: http://schema.org/FailedActionStatus
- [File]: http://schema.org/MediaObject
- [Journal]: http://schema.org/Periodical
- [GeoCoordinates]: http://schema.org/GeoCoordinates
- [ImageObject]: http://schema.org/ImageObject
- [MediaObject]: http://schema.org/MediaObject
- [Organization]: http://schema.org/Organization
- [Person]: http://schema.org/Person
- [PotentialActionStatus]: http://schema.org/PotentialActionStatus
- [Place]: http://schema.org/Place
- [Product]: http://schema.org/Product
- [Property]: http://schema.org/Property
- [PropertyValue]: http://schema.org/PropertyValue
- [ScholarlyArticle]: http://schema.org/ScholarlyArticle
- [SoftwareApplication]: http://schema.org/SoftwareApplication
- [SoftwareSourceCode]: http://schema.org/SoftwareSourceCode
- [UpdateAction]: http://schema.org/UpdateAction
- [WebSite]: http://schema.org/WebSite

- [about]: http://schema.org/about
- [accountablePerson]: http://schema.org/accountablePerson
- [actionStatus]: http://schema.org/actionStatus
- [additionalType]: http://schema.org/additionalType
- [affiliation]: http://schema.org/affiliation
- [agent]: http://schema.org/agent
- [alternateName]: http://schema.org/alternateName
- [author]: http://schema.org/author
- [citation]: http://schema.org/citation
- [contact]: http://schema.org/accountablePerson
- [contactPoint]: http://schema.org/contactPoint
- [contactType]: http://schema.org/contactType
- [contentLocation]: http://schema.org/contentLocation
- [contentSize]: http://schema.org/contentSize
- [contributor]: http://schema.org/contributor
- [copyrightHolder]: http://schema.org/copyrightHolder
- [creator]: http://schema.org/creator
- [dateCreated]: http://schema.org/dateCreated
- [datePublished]: http://schema.org/datePublished
- [defaultValue]: http://schema.org/defaultValue
- [description]: http://schema.org/description
- [distribution]: http://schema.org/distribution
- [email]: http://schema.org/email
- [encodingFormat]: http://schema.org/encodingFormat
- [endTime]: http://schema.org/endTime
- [error]: http://schema.org/error
- [event]: http://schema.org/event
- [familyName]: http://schema.org/familyName
- [funder]: http://schema.org/funder
- [geo]: http://schema.org/geo
- [givenName]: http://schema.org/givenName
- [hasPart]: http://schema.org/hasPart
- [identifier]: http://schema.org/identifier

- [IndividualProduct]: http://schema.org/IndividualProduct
- [instrument]: http://schema.org/instrument
- [keywords]: http://schema.org/keywords
- [license]: http://schema.org/license
- [memberOf]: http://schema.org/memberOf
- [name]: http://schema.org/name
- [object]: http://schema.org/object
- [phone]: http://schema.org/phone
- [programmingLanguage]: http://schema.org/programmingLanguage
- [publisher]: http://schema.org/publisher
- [relatedItem]: http://schema.org/relatedItem
- [result]: http://schema.org/result
- [sameAs]: http://schema.org/sameAs
- [sdLicense]: http://schema.org/sdLicense
- [sdPublisher]: http://schema.org/sdPublisher
- [sdDatePublished]: https://schema.org/sdDatePublished
- [startTime]: http://schema.org/startTime
- [temporalCoverage]: http://schema.org/temporalCoverage
- [thumbnail]: http://schema.org/thumbnail
- [translationOf]: http://schema.org/translationOf
- [translator]: http://schema.org/translator
- [url]: http://schema.org/url
- [valueRequired]: http://schema.org/valueRequired
- [version]: http://schema.org/version
- [creativeWorkStatus]: http://schema.org/creativeWorkStatus
- [funding]: https://github.com/schemaorg/schemaorg/pull/2618
- [isBasedOn]: http://schema.org/isBasedOn
- [maintainer]: http://schema.org/maintainer
- [producer]: http://schema.org/producer
- [runtimePlatform]: http://schema.org/runtimePlatform
- [softwareRequirements]: http://schema.org/softwareRequirements
- [targetProduct]: http://schema.org/targetProduct
- [conditionsOfAccess]: http://schema.org/conditionsOfAccess

- [dateModified]: http://schema.org/dateModified
- [image]: http://schema.org/image
- [Grant]: http://schema.org/Grant
- [Project]: http://schema.org/Project
- [subjectOf]: http://schema.org/subjectOf
- [mainEntityOfPage]: http://schema.org/mainEntityOfPage
- [WebPage]: https://schema.org/WebPage
- [input]: https://bioschemas.org/types/ComputationalWorkflow/0.1-DRAFT-2020_07_21/#input
- [output]: https://bioschemas.org/types/ComputationalWorkflow/0.1-DRAFT-2020_07_21/#output
- [FormalParameter]: https://bioschemas.org/types/FormalParameter/0.1-DRAFT-2020_07_21/
- [ComputationalWorkflow]: https://bioschemas.org/types/ComputationalWorkflow/0.1-DRAFT-2020_07_21/
- [ComputationalWorkflow profile]: https://bioschemas.org/profiles/ComputationalWorkflow/0.5-DRAFT-2020_07_21/
- [FormalParameter profile]: https://bioschemas.org/profiles/FormalParameter/0.1-DRAFT-2020_07_21/
- [RFC 2119]: https://tools.ietf.org/html/rfc2119
- [RFC 3986]: https://tools.ietf.org/html/rfc3986
- [RFC 6838]: https://tools.ietf.org/html/rfc6838
- [RFC 7159]: https://tools.ietf.org/html/rfc7159
- [RFC 8493]: https://tools.ietf.org/html/rfc8493