# Notes on JSON-LD implementation for documenting ECRR resources

Stephen M. Richard
EarthCube GeoCODES team
April 26, 2022
Current version: Google Doc

# Introduction

This is a description of the EarthCube Resource Registry metadata schema for describing a variety of information resources useful for geoscience research and data managment. The document first discusses the fields in the schema that apply to all resources, followed by a section that documents metadata fields and conventions specific to the various resource types in the scope of the resource registry. In each section, the mandatory fields are described first, highlighted in red. Other fields are described in alphabetic order by name. A final section describes some general implementation patterns used in the JSON LD encoding.

For more information on the design and implementation of the registry see Final Report EarthCube Resource Registry Implementation (https://doi.org/10.5281/zenodo.3840744).

# Resource description properties

## Mandatory for all resource

Required information:

## MetadataIdentifier

- JSON key: '@id'.
- Value: string that is unique in the scope of the containing repository
- An identifier string for this metadata record, commonly a UUID with or without an HTTP prefix for web resolution. NOTE that the identifier for the metadata record (this identifier) is expected to be unique, and different from the identifier for the described resource. If record identifiers are provided by a harvested data publisher, they must be checked for uniqueness in the aggregating metadata repository. In general these identifiers are automatically generated opaque identifiers assigned by metadata creation software.

## JSON-LD Object Type

- JSON key: '@type'.
- Value: Class name from schema.org vocabulary; expected values are {CreativeWork, SoftwareApplication, SoftwareSourceCode, Product, WebAPI}
- A schema.org Class (subclass of schema.org Thing) that specifies the expected information content for the metadata record. The type classes for a given resource description are selected based on the schema.org properties used to describe the resource such that the record will validate with the schema.org validation tool.

## Resource Name

- JSON key: 'name'
- Value: text
- Short name by which this resource will be recognized; for human users to identify the resource; should be unique in the scope of the ECRR registry and informative enough to tell someone what the described resource is.

## Description

- JSON key: 'description'
- Value: text (100 characters minimum)
- A text description of the resource. This text will be indexed by search aggregators, and the information contained should be sufficient to tell a person what the resource is, broadly how to use it. Should provide as much detail as possible, so that search engine text indexing will provide useful results. Feel free to copy and paste from respective web sites, papers, reports, etc.

## License

- JSON key: 'license'
- Values: Array of labeled link objects, implemented as CreativeWork; range is a controlled vocabulary of common licenses.
- This property identifies the statement of conditions for use and access to the described resource. Note that a resource may be available under more than one license.

Table 1. Licenses

| License Name | URI |
|---|---|
| Apache | http://cor.esipfed.org/ont/SWL_0000013 |
| BSD | http://cor.esipfed.org/ont/SWL_0000015 |
| Common Development and Distribution License 1.0 | https://opensource.org/licenses/CDDL-1.0 |
| Creative Commons Attribution (CC BY) | http://cor.esipfed.org/ont/CCL_0000001" |
| Creative Commons Attribution ShareAlike (CC BY-SA) | http://cor.esipfed.org/ont/CCL_0000002" |
| Creative Commons Attribution-NoDerivatives 1.0 | http://cor.esipfed.org/ont/CCL_0000003 |
| Creative Commons Attribution-NonCommercial (CC BY-NC) | http://cor.esipfed.org/ont/CCL_0000004 |
| Creative Commons Attribution-NonCommercial-NoDerivatives (CC BY-NC-ND) | http://cor.esipfed.org/ont/CCL_0000007 |
| Creative Commons Attribution-NonCommercial-ShareAlike (CC BY-NC-SA) | http://cor.esipfed.org/ont/CCL_0000005 |
| Creative Commons CC0 'No Rights Reserved' | https://creativecommons.org/publicdomain/zero/1.0/ |
| Creative Commons PDM 'No Known Copyright' | http://core.esipfed.org/ont/SWL_0000014 |
| Creative Commons Public Domain | http://cor.esipfed.org/ont/CCL_0000011 |
| Eclipse Public License version 2.0 | https://opensource.org/licenses/EPL-2.0 |
| GNU Affero General Public License (AGPL) | https://www.gnu.org/licenses/agpl-3.0.html |
| GNU General Public License (GPL) | http://cor.esipfed.org/ont/SWL_0000017 |
| GNU Lesser General Public License (LGPL) | http://cor.esipfed.org/ont/SWL_0000018 |
| MIT | https://opensource.org/licenses/MIT |
| Mozilla Public License 2.0 (MPL-2.0) | https://opensource.org/licenses/MPL-2.0 |
| Proprietary | http://cor.esipfed.org/ont/SWL_0000019 |

Example:

```
"license": [
        {   "@type": "CreativeWork",
            "name": "MIT",
            "identifier": "https://opensource.org/licenses/MIT"
        }  ],
```

## Type of Resource

- JSON key: mainEntity
- Value: array of labeled link objects as CreativeWork; range is a controlled vocabulary, resource types defined in ECRR ontology for resources, semantic resources, and specification types. (Table 2, 3, 4)
- This property identifies the EarthCube specific resource type. The name in the labeled link object is the ECRR resource type name. The ECRR resource type (Table 2) is used to validate resource specific properties. A resource type from Table 2 is required; if the resource type is Specification or Semantic Resource, a more specific resource type from Table 3 or Table 4 (respectively) can be specified.

Table 2. ECRR resource types.

| **Label** | **URI** |
|---|---|
| Bundled Object | http://cor.esipfed.org/ont/earthcube/SFO_0000075 |
| Catalog/Registry | http://cor.esipfed.org/ont/earthcube/ECRRO_0000212 |
| Interchange file format | http://cor.esipfed.org/ont/earthcube/ECRRO_0000208 |
| Interface/API | http://cor.esipfed.org/ont/earthcube/ECRRO_0000207 |
| Platform | http://cor.esipfed.org/ont/earthcube/ECRRO_0000211 |
| Repository | http://cor.esipfed.org/ont/earthcube/ECRRO_0000209 |
| Semantic Resource | http://cor.esipfed.org/ont/earthcube/ECRRO_0000210 |
| Service Instance | http://cor.esipfed.org/ont/earthcube/ECRRO_0000202 |
| Software | http://cor.esipfed.org/ont/earthcube/ECRRO_0000206 |
| Specification | http://cor.esipfed.org/ont/earthcube/ECRRO_0000204 |
| Use Case | http://cor.esipfed.org/ont/earthcube/ECRRO_0000005 |

Example:

```
"mainEntity":[{
        "@type": "CreativeWork",
        "url": "http://cor.esipfed.org/ont/earthcube/ECRRO_0000206",
        "name": "Software"
        }  ],
```

Specification and Semantic Resource resource types have more specific categories that are serialized as additional Main Entity values.

Table 3. More specific resource subclasses of Specification.

| **Label** | **URI** |
|---|---|
| API | http://cor.esipfed.org/ont/earthcube/SPKT_0000012 |
| Data Conversion | http://cor.esipfed.org/ont/earthcube/SPKT_0000008 |
| Data Format Convention | http://cor.esipfed.org/ont/earthcube/SPKT_0000001 |
| Data Model | http://cor.esipfed.org/ont/earthcube/SPKT_0000004 |
| Data Model Convention | http://cor.esipfed.org/ont/earthcube/SPKT_0000005 |
| File Packaging Convention | http://cor.esipfed.org/ont/earthcube/SPKT_0000010 |
| Interoperability Specification | http://cor.esipfed.org/ont/earthcube/SPKT_0000011 |
| Metadata Convention | http://cor.esipfed.org/ont/earthcube/SPKT_0000006 |
| Metadata Vocabulary | http://cor.esipfed.org/ont/earthcube/SPKT_0000009 |
| Naming Convention | http://cor.esipfed.org/ont/earthcube/SPKT_0000002 |
| Process Model | http://cor.esipfed.org/ont/earthcube/SPKT_0000013 |
| Programming Language | http://cor.esipfed.org/ont/earthcube/SPKT_0000007 |
| Web Service Convention | http://cor.esipfed.org/ont/earthcube/SPKT_0000003 |

Table 4. Mores specific subclasses of Semantic Resource.

| **Label** | **URI** |
|---|---|
| Conceptual Model | http://cor.esipfed.org/ont/earthcube/srt_0000006 |
| Controlled Vocabulary | http://cor.esipfed.org/ont/earthcube/srt_0000004 |
| Glossary | http://cor.esipfed.org/ont/earthcube/srt_0000001 |
| Ontology | http://cor.esipfed.org/ont/earthcube/srt_0000003 |

| Label | URI |
|---|---|
| RDF Vocabulary | http://cor.esipfed.org/ont/earthcube/srt_0000007 |
| SKOS Vocabulary | http://cor.esipfed.org/ont/earthcube/srt_0000008 |
| Taxonomy | http://cor.esipfed.org/ont/earthcube/srt_0000005 |
| Thesaurus | http://cor.esipfed.org/ont/earthcube/srt_0000002 |

Example:

```
"mainEntity":[
      { "@type": "CreativeWork",
        "url": "http://cor.esipfed.org/ont/earthcube/ECRRO_0000210",
        "name": "Semantic Resource"
       },
      { "@type": "CreativeWork",
        "url": "http://cor.esipfed.org/ont/earthcube/srt_0000003",
        "name": "Ontology"
        }  ],
```

## Registration metadata.

- JSON key: additionalProperty
- Value: implemented with a schema:StructuredValue object
  Schema.org does not include elements for documenting the provenance of the metadata, so this work around is implemented. This requires a value object, with a name and date published. datePublished and contributor are not properties expected for schema:StructuredValue, so the Schema.org validator will throw an error. Content in this element should be populated automatically by metadata editing tools.

  Example:

```
"additionalProperty":{
   "@type": "PropertyValue",
   "propertyID": "ecrro:ECRRO_0001301",
   "name": "registration metadata",
   "value": {
      "@type": "StructuredValue",
      "additionalType": "ecrro:ECRRO_0000156",
      "contributor":
         {"@type": "Person",
          "name": "Stephen M. Richard"},
      "datePublished": "Fri Feb 12 2021 12:50:57 GMT-0700"
    }
   }
```

## Recommended Properties for all resources

### Keywords

- JSON key: keywords
- Value: string.
- Index words to guide discovery, should include words or phrases that users might to enter into search queries. A comma delimited list of keywords as a single string. Individual keywords should not contain commas.

Example:

```
"keywords": "Proteomics,ocean,microbial ecosystems,biogeochemistry",
```

### Citation

- JSON key: additionalProperty
- Value: string
- The recommended string to use for referencing this resource in publications, using a standard bibliographic format (50 characters minimum). This is implemented as a schema.org additionalProperty. The schema.org citation property documentation states that it should be used to provide references to related resources, not to recommend a citation for the resource documented by the schema.org record, which is the intention here. In ECRR metadata, such links are implemented using the 'isRelatedTo' property, see Related Resource section.

Example:

```
"additionalProperty": {
      "@type": "PropertyValue",
      "propertyID": "dc:BibliographicCitation",
      "name": "Bibliographic citation",
      "value":"ESRI, 1998-07, ESRI Shapefile Technical Description:
          Environmental Systems Research Inc, accessed 2021-01-06,
          https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf
          "  },
```

### URL to User-Readable Page

- JSON key: subjectOf
- Value: Array of labeled link objects;
- The URL in the labeled link should access a web page with information about the resource on the Web, the name is intended for use in user interfaces to present as an html anchor.

Example:

```
 "subjectOf": [
   {"@type": "CreativeWork",
    "name": "manual page",
```

```
    "url":"http://ds.iris.edu/files/sac-manual/manual/file_format.html"
  }  ],
```

## Resource Identifier

- JSON key: identifier
- Value: array of strings
- One or more globally-unique URIs, ideally using a scheme that can be dereferenced on the Web. HTTP URI's are preferred, but any URI scheme is allowed, e.g. ISSN, DOI, ARK, ISBN, URN:OGC. For resource type = 'Interchange file format', the identifier is the string used to identify the file format. ECRR uses the MIME syntax, with type properties on the MIME type to provide profile-specific format information. For some formats, a legacy identifier string is in use and is also included.

  Example 1. Identifier for Interchange format resource:

```
"identifier":
          ["application/xml;type=wqx3",
          "http://www.exchangenetwork.net/schema/wqx/3"],
```

  Example 2. Identifier for Ontology resource

```
"identifier":  ["http://cor.esipfed.org/ont/earthcube/ecrro"],
```

## Responsible Parties

- JSON key: any of creator, editor, contributor, or publisher
- Value: each key can have an array of values, each of which is an agent type defined in the ECRR schema.
- Credit can be assigned to various agents (schema:Person or schema:Organization) using the roles defined by schema.org, which include creator, editor, contributor, and publisher. Any schema:Person or schema:Organization property can be included, but schema:name is mandatory. If an identifier is available, it should be included.

  Example:

```
"creator": [
       {   "@type": "Person",
           "name": "Nicholas McKay"
       },
       {   "@type": "Person",
           "name": "Julien Emile-Geay"
       }
   ],
"editor": [
       {   "@type": "Person",
           "name": "Giulietta S. Fargion",
           "identifier":"https://orcid.org/0000-0005-3824-4100"
       },
 "publisher": [
```

```
        {   "@type": "Organization",
            "name": "Incorporated Research Institutions for
               Seismology (IRIS)",
            "identifier": "https://ror.org/05xkn9s74"
        }
    ],
```

## Other Properties for all resources

### Alternate Resource Name(s)

- JSON key: alternateName
- Value: array of strings
- Other names by which the resource might be known or discovered. If names are provided in a non-English language, please suffix a language identifier using the '@' delimiter and an ISO-639-1 two letter language code, e.g.

  Example:
```
  "alternateName": ["WMO Binary Universal Form",
                    "Forme universelle binaire de l'OMM@fr"].
```

### Primary Publication

- JSON key: additionalProperty
- Value: schema:PropertyValue//string
- This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000600. The value should be either an identifier (e.g. DOI) or standard academic citation for publication that specifies or describes the resource.

  Example:
```
"additionalProperty": {
    "@type": "PropertyValue",
    "propertyID": "ecrro:ECRRO_0000600",
    "name": "primary publication",
    "value":"Barnes, Stanley L., 1980, Report on a Meeting to
      Establish a Common Doppler Radar Data Exchange Format:
      Bulletin of the American Meteorological Society, vol. 61, no.
      11, pp. 1401-1404. (accessed at
      http://www.jstor.org/stable/26221476)"
  },
```

### Maturity/Status

- JSON key: additionalProperty
- Value: schema:PropertyValue//schema:DefinedTerm

- This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000138. The value indicates the degree to which the resource has been tested and validated for accuracy and conformance to applicable specifications. Values are Maturity assessment category terms from the ECRR controlled vocabulary (http://cor.esipfed.org/ont/earthcube/MTU), and are implemented as schema:DefinedTerm so that a label and URI for the category can be included. This is valid JSON, but not consistent with the schema.org expected data types for PropertyValue/value, which are Boolean, Number, StructuredValue, or Text. Conceptually the DefinedTerm is a structured value.

  Example:

```
"additionalProperty": {
    "@type": "PropertyValue",
    "propertyID": "ecrro:ECRRO_0000138",
    "name": "has maturity state",
    "value": {
        "@type": "DefinedTerm",
        "name": "Used in multiple places",
        "identifier":
            "http://cor.esipfed.org/ont/earthcube/MTU_0000001"
     } }
```

## Expected Lifetime

- JSON key: additionalProperty
- Value: schema:PropertyValue//schema:DefinedTerm
- This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000219. This property specifies how long is it anticipated that the resource will be maintained and accessible online. Values are from the ECRR Expected Lifetime Controlled Vocabulary (http://cor.esipfed.org/ont/earthcube/ELT) and are implemented as schema:DefinedTerm so that a label and URI for the category can be included. This is valid JSON, but not consistent with the schema.org expected data types for PropertyValue/value, which are Boolean, Number, StructuredValue, or Text. Conceptually the DefinedTerm is a structured value.

  Example:

```
"additionalProperty": {
    "@type": "PropertyValue",
    "propertyID": "ecrro:ECRRO_0000219",
    "name": "expected lifetime",
    "value": {
        "@type": "DefinedTerm",
        "name": "1 - 5 years",
```

```
            "identifier":
                "http://cor.esipfed.org/ont/earthcube/ELT_0000003"
        } }
```

## Version of the Resource

- JSON key: version
- Value: string
- This property identifies a particular version of the resource if it is not identified by the schema:identifier element.

  Example:
  ```
  "version":"0.6.2",
  ```

## Funding

This property specifies the source or sources of financial support for the creation and maintenance of the described resource. Note that the current implementation is based on schema.org entities defined as of 2018. The proposed addition of a new property 'funding' was adopted by schema.org in March, 2022. This section includes a description of the existing implementation, and the new recommended implementation using the 'funding' property.

Existing implementation:

- JSON key: funder
- Value: array of either Person, Organization, or Grant
- Person or Organization: Required name, recommended but optional identifier. Other person or organization properties could be added
  Grant: Required name of grant, and identifier if applicable/available, and the funding organization implemented as schema:FundingAgency (a subclass of Project). FundingAgency requires a name, and a recommended but optional identifier for the organization.

  Example:
  ```
  "funder" : [
          {   "@type": "Organization",
              "name": "NOAA Coastal Services Center (NOAA-CSC)"
          },
          {   "@type": "Grant",
              "name": "Award ICER 1541008",
              "funder": {
                  "@type": "FundingAgency",
                  "name": "US NSF",
                  "identifier": "https://ror.org/021nxhr62"
              }
          }  ],
  ```

New (post 03/2022) implementation:

```
      "funding" : [
```

```
{   "@type": "Grant",
    "sponsor": {
        "@type": "Organization",
        "name": "NOAA Coastal Services Center (NOAA-CSC)"
    }  },
{"@type": "Grant",
    "name": "Award ICER 1541008",
    "funder": {
        "@type": "Organization",
        "name": "US NSF",
        "identifier": "https://ror.org/021nxhr62"
    }  }  ],
```

## Stewardship property:

- JSON key: additionalProperty
- Value: schema:PropertyValue//schema:DefinedTerm
- This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000218. Could be person or organization, with just a name string, a name string and identifier, or more information. For consistency, always make value an object.

  Example Person:

```
"additionalProperty":
      {"@type": "PropertyValue",
        "propertyID": "ecrro:ECRRO_0000218",
        "name": "Stewardship",
        "value": {"@type": "Person",
                  "name": "Ben Best",
                  "url": "https://orcid.org/0000-0002-2686-0784"
                  }
            },
```

  Example Organization:

```
"additionalProperty":
      { "@type": "PropertyValue",
        "propertyID": "ecrro:ECRRO_0000218",
         "name": "Stewardship",
         "value":
            {"@type":"Organization",
             "name":"Frictionless Data Github Organization",
             "url":"https://frictionlessdata.io/"}
      },
```

## Intended Audience/Target User

- JSON key: audience

- Value: array of schema:Audience
- Term to specify the type of agent intended to utilize the resource. Terms from ECRR controlled vocabulary (http://cor.esipfed.org/ont/earthcube/AUT) to identify the kinds of users who are the target of the described resource.

  Example:

```
"audience": [
   {
      "@type": "Audience",
      "audienceType": "Data Producers",
      "identifier":
         "http://cor.esipfed.org/ont/earthcube/AUT_0000001"
      },
   {
      "@type": "Audience",
      "audienceType": "Data Users",
      "identifier":
         "http://cor.esipfed.org/ont/earthcube/AUT_0000002"
      },
    {
      "@type": "Audience",
      "audienceType": "Data Facilities and Repositories",
      "identifier":
         "http://cor.esipfed.org/ont/earthcube/AUT_0000003"
        } ],
```

## Usage

- JSON key: additionalProperty
- Value: schema:PropertyValue//schema:DefinedTerm
- Specifies qualitatively how much the resource is being used in the community (at the time the metadata record was created or updated). This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000017. Values are from ECRR Controlled Vocabulary (http://cor.esipfed.org/ont/earthcube/UBA) and are implemented as schema:DefinedTerm so that a label and URI for the category can be included. This is valid JSON, but not consistent with the schema.org expected data types for PropertyValue/value, which are Boolean, Number, StructuredValue, or Text. Conceptually the DefinedTerm is a structured value.

  Example:

```
"additionalProperty":  {
      "@type": "PropertyValue",
      "propertyID": "ecrro:ECRRO_0000017",
      "name": "Usage",
      "value": {
          "@type": "DefinedTerm",
          "name": "Some usage (10-50 adopters)",
```

```
            "identifier":
                "http://cor.esipfed.org/ont/earthcube/UBA_0000002"
        }  },
```

## Science Domain

- JSON key: about
- Value: array of schema:DefinedTerm
- Specifies the science disciplines related to the described resource. Values are from ECRR Academic Disciplines Controlled Vocabulary (http://cor.esipfed.org/ont/earthcube/ADO) and are implemented as schema:DefinedTerm so that a label and URI for the category can be included.

  Example:

```
"about": [
        {   "@type": "DefinedTerm",
            "name": "Geodynamics",
            "identifier":
                 "http://cor.esipfed.org/ont/earthcube/ADO_0000093"
        },
        { "@type": "DefinedTerm",
            "name": "Geomagnetism",
            "identifier":
                 "http://cor.esipfed.org/ont/earthcube/ADO_0000094"
        },
        {  "@type": "DefinedTerm",
            "name": "Paleomagnetism",
            "identifier":
                 "http://cor.esipfed.org/ont/earthcube/ADO_0000098"
        } ],
```

## Dependencies

- JSON key: additional Property
- Value: array of schema:CreativeWork
- Specifies other resources that must be accessible or generate output required by the described resource. This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://purl.obolibrary.org/obo/RO_0002502. Values are implemented as schema:CreativeWork, with a required name property and recommended URL property

  Example:

```
"additionalProperty":
        {"@type": "PropertyValue",
         "propertyID": "http://purl.obolibrary.org/obo/RO_0002502",
         "name": "dependencies",
         "Value": [
           { "@type": "CreativeWork",
             "name": "Java 1.8 Runtime Environment (JRE)",
```

```
                "url":
                      "http://www.oracle.com/technetwork/java/javase/do
                      wnloads/jre8-downloads-2133155.html"  },
              {"@type":"CreativeWork",
               "name":"osgeo, ipywidgets, requests, json,
                       pandas.io.json"}
          ] },
```

### Related Resources

- JSON key: isRelatedTo
- Value: array of JSON objects
- List of schema.org types, names and URLs for other resources of interest related to the resource. The default type is CreativeWork, but a resource can be related to any kind of schema.org entity that makes sense. The expected domain for schema:isRelatedTo is schema:Productor schema:Service, thus the @type in the root of the metadata record MUST include schema:Product (or Service) to validate. Value objects are required to have a name property and a recommended URL property. Other properties consistent with the defined @type can be included, for instance schema:description could be used to document the relationship to the linked resource. [TBD-- investigate used of schema:LinkRole for this use case].

  Example:

```
"isRelatedTo": [
        {   "@type": "CreativeWork",
            "name": "Ocean Data View (ODV)",
            "url": "https://odv.awi.de/"
        }  ],
```

## Resource specific recommendations

This section includes discussions on recommendations for representing the various kinds of resources in the scope of the Earth Cube Resource Registry. These recommendations include fields that are specific to particular resource types, as well as conventions for the meaning of content in fields that are used for more than one resource.

### Semantic resource

This resource type includes data models or vocabularies that define concepts and optionally relationships representing a conceptualization of some domain of discourse. Includes a spectrum of resources including (not limited to!) controlled vocabulary, thesaurus, information model, taxonomy, glossary, and vocabulary. For linked data purposes, it is useful to specify the entities and properties of the entities represented by the model using resolvable URI references to widely used ontologies. Specific properties are ResourceType (required), Model Language, and Representation Format.

ResourceType

- JSON key: mainEntity
- Value: array of labeled links (schema:CreativeWork)

Semantic resources have a more granular resource type classification to provide better information about what kind of resource it is. See Table 4 above. Thus, a semantic resource will have a mainEntity that is Semantic Resource and a second main entity value that is the specific kind of semantic resource from the ECRR semantic resource type vocabulary (http://cor.esipfed.org/ont/earthcube/srt).

Example:

```
"mainEntity":[
      { "@type": "CreativeWork",
        "url": "http://cor.esipfed.org/ont/earthcube/ECRRO_0000210",
        "name": "Semantic Resource"
        },
      { "@type": "CreativeWork",
        "url": "http://cor.esipfed.org/ont/earthcube/srt_0000007",
        "name": "RDF Vocabulary"
      }  ],
```

Model language

- JSON key: programmingLanguage
- Value: array of schema:ComputerLanguage instances
  This property specifies that formal language used to represent the content of the semantic resource, typically OWL, RDF, or SKOS. name is required.

Example:

```
"programmingLanguage": [
        {   "@type": "ComputerLanguage",
            "name": "OWL",
            "identifier":
                 "http://cor.esipfed.org/ont/earthcube/MOLA_0000001"
        }  ],
```

Representation format

- JSON key: encodingFormat
- Value: array of strings
  Representation Format or base format for a profile specification. Use a registered MIME type prefix for the base format, and an application specific extension (after a ';' delimiter. Values should be from the register at https://github.com/earthcube/GeoCODES-Metadata/blob/main/resources/encodingFormat.csv

Example:

```
"encodingFormat": [
        "application/xml;type=rdf",
        "text/rdf+n3",
        "text/turtle"  ],
```

## Interchange format

An interchange format is a serialization scheme (data format) for communication between agents. Formats of particular interest for the ECRR are those used to serialize data for preservation in files or transfer between software applications. In the ECRR documentation for interchange formats, the schema:identifier is the string that should be used to identify the file format for dataset distributions using this format. ECRR uses the MIME syntax, with type properties on the MIME type to provide profile-specific format information. Terms from the ECRR format registry should be used. If an existing identifier string is in use, best practice is to include that identifier in the encodingFormat array. .

### Base format

- JSON key: encodingFormat
- Value: array of string
- This property identifies the base MIME type for files encoded using the format. The identifier for the format will typically have an extension that identifies a particular implementation or profile of the format. For instance the base format might be 'application/xml', but a registered interchange format would identify an xml schema and perhaps schematron rules that define a more specific content and encoding that make the format useful for a particular application.

  Example 1, Standard MIME type only:

```
"encodingFormat": ["application/x-netcdf"],
```

  Example 2, type identifies a specific application profile; also has legacy identifier string that is the URI for the xml namespace:

```
"identifier":
            ["application/xml;type=wqx3",
            "http://www.exchangenetwork.net/schema/wqx/3"],
```

### Semantic resources used

- JSON key: isBasedOn
- Value: array of labeled links as schema:CreativeWork
- A list resources used in the definition of the interchange format, e.g. vocabularies used, or imported schema.

  Example:

```
"isBasedOn": [
        {
            "@type": "CreativeWork",
            "name": "External Data Representation Standard(XDR)",
            "url": "https://tools.ietf.org/html/rfc4506"
        }   ],
```

### Conforms to

- JSON key: dct:conformsTo
- Value: array of labeled links as schema:CreativeWork

- A list of specifications that define the interchange format. Implemented as labeled links using schema:CreativeWork.

  Example:

```
"dct:conformsTo": [
  {
      "@type": "CreativeWork",
      "name": "Frictionless Data Data Package Specification",
      "url": "https://specs.frictionlessdata.io/data-package/"
  } ],
```

## Specification

A specification is a document that defines how a particular activity can be done for consistency between different implementers of that activity. Examples of scope for a specification include interchange formats, vocabularies, programming languages, inter-application interfaces in operating systems, and web services. The Specification types listed in Table 3 define kinds of specifications encountered in the EarthCube GeoCODES community. Software applications, service instances, and interchange formats all implement a specification, whether or not it is explicitly documented in a separate citable document.

ResourceType

- JSON key: mainEntity
- Value: array of labeled links (schema:CreativeWork)
  Specifications have a more granular resource type classification to provide better information about what kind of resource it is. See Table 3, above. Thus, a semantic resource will have a mainEntity that is Specification and a second main entity value that is the specific kind of specification from the ECRR specification type vocabulary (http://cor.esipfed.org/ont/earthcube/spkt).

  Example:

```
"mainEntity":[
  {
      "@type": "CreativeWork",
      "url": "http://cor.esipfed.org/ont/earthcube/ECRRO_0000204",
      "name": "Specification"
  },
  {
      "@type": "CreativeWork",
      "url": "http://cor.esipfed.org/ont/earthcube/SPKT_0000001",
      "name": "Data Format Convention"
  }  ],
```

ProfileOf

- JSON key: additionalProperty
- Value: array of labeled links as schema:CreativeWork.

A profile defines a set of clauses, classes, options or parameters from one or more base specifications that are used to implement a particular set of requirements. Profiles must be constructed such that conformance to the profile implies conformance to the base specification from which it is derived. This property is only applicable if resource type is Specification (mainEntity = ecrro:ECRRO_0000204). The value is a link to one or more base specifications for the profile. This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000501. Values are labeled links, implemented as schema:CreativeWork, with a required name property and recommended URL property.

Example:

```
"additionalProperty":{
      "@type": "PropertyValue",
      "propertyID": "ecrro:ECRRO_0000501",
      "name": "profile of",
      "value": [
          {   "@type": "CreativeWork",
              "name": "IETF 4180",
              "url":"https://www.ietf.org/rfc/rfc4180.txt"
          }
      ] },
```

## Catalog

A Catalog is a curated collection of descriptions of resources, accessible through one or more interfaces.

Content types cataloged

- JSON key: contentType
- Value: array of labeled links as schema:Thing.
  Content Type(s) of Cataloged Objects; one object or array of objects, could be anything consistent with schema:Thing; each with required name and optional identifier. Note the usage here is not consistent with schema.org definition of contentType, which limits its domain to schema:EntryPoint and range to schema:Text, so validator will throw errors.

  Example:

```
"contentType":[
       {   "@type":"Thing",
           "name":"Dataset",
           "identifier":"https://schema.org/Dataset"
       }   ],
```

## Interface or API Specification

A specification that defines a web service (webAPI) intended for implementation by multiple servers (e.g. OGC API). In the original ECRR design, Interface/API was considered as separate resource. Based on the resource descriptions compiled in the initial development phase, it

became apparent that it is unnecessary to make a distinction between the interface as a conceptual entity, the specification for that interface, and the applications/service instances that implement the interface. There are many API endpoints that are implemented with some documentation specific to that endpoint, but not a specification intended to be useful to implement other service endpoints that operate the same way. Service instances should be represented as Service Instance resources (see below), with dct:conformsTo providing a link to the service specification (this resource type) if there is such a specification.

Resource Type

- JSON key: mainEntity
- Value: array of CreativeWork
- MUST have resource type 'API' (for interfaces that might be implemented in any computational environment) or 'Web Service Convention' (for interfaces that operate via HTTP on the World Wide Web, e.g. WebAPIs), as well as resource type for 'Specification'

  Example:

```
"mainEntity": [
  { "@type": "CreativeWork",
    "url": "http://cor.esipfed.org/ont/earthcube/ECRRO_0000204",
    "name": "Specification"
  },
  { "@type": "CreativeWork",
    "url": "http://cor.esipfed.org/ont/earthcube/SPKT_0000012",
    "name": "API"
  }    ],
```

Communication protocol

- JSON key: additionalProperty
- Value: array of labeled links as schema:DefinedTerm.
- Name of the protocol used to interact through the interface. This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000502. Values are labeled links, implemented as schema:DefinedTerm, with a required name property and recommended identifier property. A standard set of protocols is defined in the ECRR communication protocols vocabulary (http://cor.esipfed.org/ont/earthcube/CMPR, Table 5), and use of terms from this vocabulary (where applicable) is recommended for interoperability.

Table 5. ECRR Communication Protocol vocabulary

| Communication Protocol | URI |
|---|---|
| HTTP | http://cor.esipfed.org/ont/earthcube/CMPR_0000001 |
| HTTPS | http://cor.esipfed.org/ont/earthcube/CMPR_0000002 |
| TCP/IP | http://cor.esipfed.org/ont/earthcube/CMPR_0000003 |

| Communication Protocol | URI |
|---|---|
| FTP | http://cor.esipfed.org/ont/earthcube/CMPR_0000004 |
| SSH | http://cor.esipfed.org/ont/earthcube/CMPR_0000005 |
| SCP | http://cor.esipfed.org/ont/earthcube/CMPR_0000006 |
| SFTP | http://cor.esipfed.org/ont/earthcube/CMPR_0000007 |
| SMTP | http://cor.esipfed.org/ont/earthcube/CMPR_0000008 |
| POP | http://cor.esipfed.org/ont/earthcube/CMPR_0000009 |

Example:

```
"additionalProperty": [
       {
           "@type": "PropertyValue",
           "propertyID": "ecrro:ECRRO_0000502",
           "name": "communication protocol",
           "value": {
              "@type": "DefinedTerm",
              "name": "TCP/IP",
              "identifier":
                 "http://cor.esipfed.org/ont/earthcube/CMPR_0000003"
            }
        } ],
```

Function

- JSON key: applicationCategory
- Value: array of string.
- The schema.org applicationCategory property has an expected value that is Text or URL. For the ECRR, a controlled vocabulary of software functions in the Earth Science research realm was compiled (http://cor.esipfed.org/ont/earthcube/sfo, Table 7 in Function section, below). String values should use this syntax: “function: ... uri: ...”. The function value is the label associated with the ECRR uri in the function vocabulary

Example:

```
"applicationCategory": [
   "function: Data Exploration uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000006",
   "function: Data Analysis uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000010"
    ],
```

Conforms to

- JSON key: dct:conformsTo
- Value: array of schema:CreativeWork

- A list of specifications that define the interface operation, including protocols used, messaging formats, operations implemented. Implemented for ECRR as labeled links using schema:CreativeWork, with required name, and recommended url.

  Example:

```
"dct:conformsTo": [
      {"@type": "CreativeWork",
       "name": "fdsnws-dataselect specification v1.1",
       "url": "http://n2t.net/ark:/23942/g2800025"
      } ],
```

## Service instance

A service instance is a particular implementation of an Interface or API Specification. In the case of WebAPI, the metadata must include a URL for the service endpoint.

URL

- JSON key: url
- Value: array of schema:LinkRole elements
- The schema:url field for a service instance should be the base url used for requests to the service. Use of LinkRole allows distinction of various links that might be provided in the url field. See example in schema.org documentation. For interoperability, the location of the service endpoint must be identified with the linkRelationship '`Service Endpoint Base URL`'

  Example:

```
"url": [
        {   "@type": "LinkRole",
            "url": "http://service.iris.edu/irisws/timeseries/1/",
            "linkRelationship": "Service Endpoint Base URL"
        }   ],
```

Interface

- JSON key: additionalProperty
- Value: array of labeled links as schema:CreativeWork.
- Links to one or more specifications that document the service implemented by this service instance. This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is http://cor.esipfed.org/ont/earthcube/ECRRO_0000503. Values are labeled links, implemented as schema:CreativeWork, with a required name property and recommended URL property. If no specification document is available, leave this property out.

  Example:

```
"additionalProperty": [
  { "@type": "PropertyValue",
    "propertyID": "ecrro:ECRRO_0000503",
    "name": "Interface specification",
```

```
    "value":[
      { "@type":"CreativeWork",
        "name": "ePandda API specification",
        "identifier":"http://n2t.net/ark:/23942/g2805001"}]
  } ],
```

Function

- JSON key: applicationCategory
- Value: array of string.
- This property specifies the kinds of activities supported by the service interface. The schema.org applicationCategory property has an expected value that is Text or URL. For the ECRR, a controlled vocabulary of software functions in the Earth Science research realm was compiled (http://cor.esipfed.org/ont/earthcube/sfo, Table 7 in Function section, below). String values should use this syntax: “function: ... uri: ...”. The function value is the label associated with the ECRR URI in the function vocabulary

  Example:

```
"applicationCategory": [
   "function: Data Exploration uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000006",
   "function: Data Analysis uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000010"
    ],
```

Machine-readable endpoint

- JSON key: isRelatedTo
- Value: array of Product.
- This property has no direct schema.org implementation, so it is implemented as a link to a related resource. The range expected for the schema:isRelatedTo property is schema:Product. A name and URL are required. Name default is ‘Machine-readable endpoint’.

  Example:

```
"isRelatedTo": [
  { "@type": "Product",
    "name": "Machine-readable endpoint",
    "url":
        "http://service.iris.edu/fdsnws/station/1/application.wadl"
        } ],
```

Potential action

- JSON key: potentialAction
- Value: array of schema:Action elements (or subtypes)
- a Schema:potentialAction, with expected value schema:Action, and the action (typically HTTP GET for a web application) is invoked via a url template specified in the Action/target/EntryPoint. If a dataset can be passed as an argument in the url, it should be indicated in the template with the template parameter ‘contentURL’. For example:

```
"urlTemplate": "https://lipd.net/playground?source={contentURL}"
```
For more information and an example, see discussion of Actions, below.

Conforms to

- JSON key: dct:conformsTo
- Value: array of schema:CreativeWork
- A list of one or more specifications that define the service operation. Implemented as labeled links using schema:CreativeWork.

  Example:
```
"dct:conformsTo": [
  {   "@type": "CreativeWork",
      "name": "Frictionless Data Data Package Specification",
      "url": "https://specs.frictionlessdata.io/data-package/"
  } ],
```

## Software:

A packaged set of instructions that can be executed by a machine to perform one or more functions. Distribution might be as source code, an executable file, an installer package, as a deployable container (e.g. Docker container) or as a web application that can be invoked via URL. In the ECRR, software resources are distinguished from Service resources in that when software is invoked via URL (Web application), the user enters an environment created by the software; when a service is invoked, the user waits for response from service and uses that in the environment from which the service was invoked.

Software can be registered at different granularity. For example a web application (e.g. GeoNetwork Opensource) can be registered as a generic software instance, with distribution linked to the source code, or as a specific geonetwork instance that hosts a particular metadata catalog, with a distribution linked to the online web application for that catalog. A Jupyter notebook can be registered as software with a link to the .ipynb file in a GitHub, but might also be registered as an instance that is accessible in a particular online Hub that can execute the notebook in a virtual environment.

Software resource descriptions are identified with the following schema.org @type and mainEntity:
```
"@type": ["CreativeWork","SoftwareApplication",
"SoftwareSourceCode"],
"mainEntity": [
  {  "@type": "CreativeWork",
     "url": "ecrro:ECRRO_0000206",
     "name": "Software"  }  ],
```
The schema:SoftwareApplication and schema:SoftwareSourceCode types are necessary to utilize some software-specific properties defined by schema.org.

Function

- JSON key: applicationCategory
- Value: array of string.

- A list of activities supported by the software. The schema.org applicationCategory property has an expected value that is Text or URL. For the ECRR, a controlled vocabulary of software functions in the Earth Science research realm was compiled (http://cor.esipfed.org/ont/earthcube/sfo, Table 7 in Function section, below). String values should use this syntax: “function: ... uri: ...”. The function value is the label associated with the ECRR uri in the function vocabulary

  Example:

```
"applicationCategory": [
   "function: Data Preparation uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000007",
   "function: Visualization uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000011",
   "function: Data Analysis uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000010",
   "function: Data Exploration uri:
              http://cor.esipfed.org/ont/earthcube/SFO_0000006"
    ],
```

Runtime environment

- JSON key: runtimePlatform
- Value: array of string.
- The hardware/operating system environment required for execution of the application. Values should be taken from the ECRR runtime environments vocabulary http://cor.esipfed.org/ont/earthcube/RTE (Table 6). The current implementation uses strings that consist of the platform name and URI. (TBD-- implement as schema: DefinedTerm)

Table 6. ECRR Runtime Environment vocabulary

| Runtime Environment | URI | Definition |
|---|---|---|
| Android | http://cor.esipfed.org/ont/earthcube/RTE_000001 | A runtime environment based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. |
| HPC | http://cor.esipfed.org/ont/earthcube/RTE_000002 | A runtime environment that runs on super computers and uses parallel processing techniques to solve complex computational problems. |
| In-the-browser | http://cor.esipfed.org/ont/earthcube/RTE_000003 | A runtime environment that executes in a web browser sandbox, using the browser for user interaction. |

| Runtime Environment | URI | Definition |
|---|---|---|
| iOS | http://cor.esipfed.org/ont/earthcube/RTE_000004 | An runtime environment created and developed by Apple Inc. exclusively for its hardware that presently powers many of the company's mobile devices, including the iPhone, and iPod Touch. |
| Linux | http://cor.esipfed.org/ont/earthcube/RTE_000005 | A family of open source Unix-like runtime environments based on the Linux kernel. |
| MacOS | http://cor.esipfed.org/ont/earthcube/RTE_000006 | A family of runtime environments developed and marketed by Apple Inc. as the primary operating system for Apple's Mac family of computers. |
| Unix | http://cor.esipfed.org/ont/earthcube/RTE_000007 | A family of multitasking, multiuser computer runtime environments that derive from the original AT&T Unix. |
| Windows | http://cor.esipfed.org/ont/earthcube/RTE_000008 | A group of graphical runtime environments, all of which are developed, marketed and sold by Microsoft, where each family caters to a certain sector of the computing industry. |
| iPadOS | http://cor.esipfed.org/ont/earthcube/RTE_000009 | A runtime environment created and developed by Apple Inc. for their iPad line of tablet computers. |
| Java Servlet Container | http://cor.esipfed.org/ont/earthcube/RTE_000010 | A runtime environment that allows an application to run as a servlet inside a software container as a component in a web server. |
| Notebook Container | http://cor.esipfed.org/ont/earthcube/RTE_000011 | Jupyter hub, etc. |

Example:

```
"runtimePlatform": [
  "Linux, ecrro:RTE_000005",
  "Windows, uri ecrro:RTE_000008"
    ],
```

Implementation language

- JSON key: programmingLanguage
- Value: array of schema:ComputerLanguage.
- The programming language or languages used to implement the application. Name is required, identifier for the software is recommended. The name (and identifier if applicable) should uniquely identify the particular version of the software described.

Example:

```
"programmingLanguage": [
     {"@type": "ComputerLanguage",
      "name": "Java 8" },
     {"@type": "ComputerLanguage",
      "name": "C++"   }],
```

Input and Output format

- JSON key: supportingData
- Value: schema:DataFeed with array of strings in the encodingFormat..
- Input and output data formats are specified using the schema:supportingData property. In general, an application does not work with one specific dataset, but if that were the case it could be specified here. The supportingData property is used to specify both input and output data, indicated by the position property, which has a range of ‘input’ or ‘output’.
  A common situation is an application that uses or produces data conforming to a particular MIME type. MIME types are mostly related to basic serialization schemes. For the dataset-application mapping use cases of particular interest for geoCODES, matching is typically based on a particular data distribution Interchange Format, which encompasses an information model and a serialization scheme. The schema:encodingFormat is used to specify the data format. For interoperability and data-application linking, using format identification strings from the ECRR formats register (https://github.com/earthcube/GeoCODES-Metadata/blob/main/resources/encodingFormat.csv) is recommended. Some interchange formats have existing identifiers that can be included as well. For instance xml interchange formats can be specified with the namespace URI for their XML schema. For data-application matching, the input encodingFormat for the software is used to match to the distribution encodingFormat for a dataset.

  Example:

```
"supportingData": [
  {"@type": "DataFeed",
   "name": "Input Data Type specification",
   "position": "input",
   "encodingFormat": [
        "application/zip;type=LiPD",
"http://linked.earth/ontology/core/1.2.0/index-en.html#Dataset",
        "application/json;type=pyleoclim",
        "text/csv;application=pyleoclim"
    ]   },
   {"@type": "DataFeed",
    "name": "Output Data Type specification",
    "position": "output",
    "encodingFormat": [
```

```
        "Multipart/Related; type=shapefile",
        "application/vnd.esri-shapefile",
        "http://www.opengis.net/doc/CS/las/1.4",
        "application/octet-stream;type=ASPRS-LAS",
        "text/plain; application=esri-asciigrid",
         "application/vnd.esri-asciigrid"
            ] }   ],
```

Code Repository

- JSON key: codeRepository
- Value: array of labeled links as schema:CreativeWork.
- URL location for repository in which code for an application or semantic resource is managed. Need a labeled link to distinguish repositories that might hold parts of the code base, or code for different target platforms. TBD: should use schema:LinkRole instead of CreativeWork; schema:codeRepository expects a URL

  Example:

```
"codeRepository": [
        {"@type": "CreativeWork",
         "name": "GeoCODES ECRR github",
         "url": "https://github.com/earthcube/ResourceRegistry"
        },
        {"@type": "CreativeWork",
         "name": "ECRR ontology github",
         "url": "https://github.com/earthcube/ecrro"
        },
        {"@type": "CreativeWork",
         "name": "Legacy developement github",
         "url":
"https://github.com/earthcubearchitecture-ecresourcereg/ecrro"
        }
    ],
```

Interface

- JSON key: additionalProperty
- Value: array of labeled links as schema:CreativeWork.
- Specifies the interface or interfaces implemented by a software application. This property has no schema.org implementation, so it is implemented in the schema:additionalProperty array. The identifier for the property is ecrro:ECRRO_0000503. Values are labeled links, implemented as schema:CreativeWork, with a required name property and recommended URL property.

  Example:

```
"additionalProperty": [
{"@type": "PropertyValue",
 "propertyID": "ecrro:ECRRO_0000503",
 "name": "Interface specification",
```

```
 "value": [
    {"@type": "CreativeWork",
     "name": "epandda data discovery API",
     "identifier": "http://n2t.net/ark:/23942/g2805001"
    } ]
} ]
```

Installer package

- JSON key: installUrl (note capitalization!)
- Value: array of labeled links as schema:CreativeWork.
- Labeled links are used to distinguish installer packages for different environments (Linux, Windows, iOS...), or different builds of the application. The url is required, name is optional. Expected type for schema:installUrl is URL, so this implementation will cause and error with the schema.org validator. TBD- implement with schema:LinkRole.

  Example:

```
"installUrl": [
   {"@type": "CreativeWork",
    "name": "unix/linux",
    "url": "http://www.geomapapp.org/UnixInstall.html"
   },
   {"@type": "CreativeWork",
    "name": "Windows",
    "url": "http://www.geomapapp.org/MSInstall.html"
    },
    {"@type": "CreativeWork",
     "name": "Mac OS",
     "url": "http://www.geomapapp.org/MacInstall.html"
    } ],
```

Web Application URL

- JSON key: potentialAction
- Value: array of schemaAction.
- Some software applications have web-native versions that can be invoked via URL. ECRR documents this kind of application execution as an Action on the Software. See the discussion and example in the PotentialAction section, below.
  Note that if a dataset can be passed as an argument in the url to invoke the application, it should indicated in the template with the template parameter ‘contentURL’.

  Example:

```
"potentialAction": [
   {"@type": "Action",
    "name": "Execute web application",
    "target": {"@type": "EntryPoint",
```

```
                "urlTemplate":
                      "https://lipd.net/playground?source={contentUR
                      L}",
                "description": "Open web application with the
                      current dataset; contentURL is expected to be
                      provided by a distribution/contentURL in
                      dataset metadata; if other parameters are
                      provided, the user will have to put those
                      in...",
                "httpMethod": ["GET"]
            }
        }
    ],
```

## Platform:

In the ECRR context, a platform is a composite software entity that enables execution of a variety of tools e.g. MatLab or ArcGIS. A key defining feature is extensibility with plugin architecture and API-based interaction between components to enable reconfiguration/extension for new problems. Platform resource descriptions are identified by @Type = [ “CreativeWork”,”SoftwareApplication”], and mainEntity with url ecrro:ECRRO_0000211.
Because the function property for the platform is implemented using applicationCategory, the schema.org type must include schema:SoftwareApplication as well as CreativeWork.

Function

- JSON key: applicationCategory
- Value: array of string.
- The schema.org applicationCategory property has an expected value that is Text or URL. For the ECRR, a controlled vocabulary of software functions in the Earth Science research realm was compiled (http://cor.esipfed.org/ont/earthcube/sfo, Table 7 in Function section, below). String values should use this syntax: “function: ... uri: ...”. The function value is the label associated with the ECRR uri in the function vocabulary

  Example:

```
"applicationCategory": [
   "function: Data Discovery & Access, uri:
http://cor.esipfed.org/ont/earthcube/SFO_0000005",
   "function: Data Analysis uri:
http://cor.esipfed.org/ont/earthcube/SFO_0000010"
    ],
```

# General Implementation patterns

## Labeled Links

Schema.org does not provide a class for a labeled link -- i.e. a URL with a text string that can be presented to users to clarify what the URL will get. The implementation approach used for the EarthCube Resource Registry JSON-LD implementation is to use the schema.org CreativeWork class, with just a name and url property. The intention is to support presenting links using html anchor elements in a web page presentation of the metadata.

Example:

```
{
"@type": "CreativeWork",
"name": "Creative Commons Attribution (CC BY)",
"url": "http://cor.esipfed.org/ont/CCL_0000001" }
```

In a web page, this would be presented as

```
<a href="http://cor.esipfed.org/ont/CCL_0000001">
      Creative Commons Attribution (CC BY)</a>
```

## Agents

The ECRR agent type is used to cite persons or organizations who play some role related to the described resource. The agent object has an @type value that is either Person or Organization. A name (string) is required, and an identifier (e.g. ORCID for person, ROR for Organization) is recommended. Other Person or Organization properties defined by schema.org can be included as well.

Example person:

```
{ "@type": "Person",
   "name": "Nicholas McKay",
   "identifier": "https://orcid.org/0000-0001-6022-8304"  },
```

Example organization:

```
{ "@type": "Organization",
 "name": "US National Science Foundation (US NSF)",
 "identifier": "https://ror.org/021nxhr62"   }
```

## Array values

If a property value is 0..*, all values will be encoded in a JSON array. This is to simplify parsing the JSON documents.

If no value is available, and the property is optional, leave it out. If the property is required, provide a value with an explicit null e.g. **urn:ogc.def.null.missing**.

## Function

Table 7. Functional categories for APIs, software, platform.

| Category | Subcategory | URI | Definition |
|---|---|---|---|
| Research Planning | | eccro: SFO_0000002 | Functions used in generating a scientific and logistic plan for executing a research project. |
| | Mental Modeling | eccro: SFO_0000015 | Functions that allow users to visually map relevant concepts, processes, and other entities relevant to planning their research. |
| | Project Planning | eccro: SFO_0000016 | Functions that allow users to develop and manage implementation of a project plan. |
| | Logistics Planning | eccro: SFO_0000017 | Functions that enable planning for the logistics needed to support research in the field. |
| | Data Management Planning | eccro: SFO_0000018 | Functions to enable the development and management of data management plans. |
| User Interaction | | eccro: SFO_0000003 | Functions that involve human-machine communication necessary for the operation of some component of a research project. |
| | Authentication & Access Control | eccro: SFO_0000019 | Functions that authenticate and/or control access to specific capabilities. |
| | User Interface | eccro: SFO_0000020 | Functions that allow a user to interact with a system. |
| | Community Support / Social Networking | eccro: SFO_0000021 | Functions that allow users to form communities across space and time and to interact with one another remotely. |
| | Notification | eccro: SFO_0000022 | Functions that allow notification of events, or other happenings. |
| | User Management | eccro: SFO_0000311 | Functions to create, update and remove user accounts and their rights and capabilities within a system. |
| Data Acquisition | | eccro: SFO_0000004 | Functions enabling incorporation of new data into an information system. Data might originate from instruments or from other information systems |
| | Instrument Control | eccro: SFO_0000023 | Functions to control the operation of a scientific instrument. |
| | Instrument Calibration | eccro: SFO_0000024 | Functions to calibrate an instrument. |

| Category | Subcategory | URI | Definition |
|---|---|---|---|
| | Streaming Data Handling | eccro: SFO_0000025 | Functions to handle data that is streaming in real-time. |
| | Data Ingest | eccro: SFO_0000026 | Functions to ingest data into a data system. |
| Data Discovery & Access | | eccro: SFO_0000005 | Functions that enable an agent to locate and access (acquire for use) information resources |
| Data Discovery & Access | Data Query | eccro: SFO_0000027 | Functions that enable an agents to query a system that retrieves a list of data matching the agent's search criteria and which might be suitable for their purpose. |
| | Data Harvesting | eccro: SFO_0000030 | The function of extracting specified information/data from specific, targeted sites. |
| | Web Crawling | eccro: SFO_0000029 | Functions that automatically and systematically search some portion of the World Wide Web for items that meet provided criteria (e.g., specific keywords, certain file types). |
| | Data Access/Order | eccro: SFO_0000028 | Functions that enable an agent to access or order data. |
| Data Exploration | | eccro: SFO_0000006 | interaction with data to determine usability, suitability for specific use, for hypothesis generation, or evaluation of patterns suggesting the need for further analysis |
| Data Exploration | Data Mining | eccro: SFO_0000033 | The function of systematically analyzing groups of data in order to uncover previously unknown patterns and relationships. |
| | Exploratory Data Analysis | eccro: SFO_0000034 | Functions that allow users to perform initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and/or to check assumptions with the help of summary statistics and graphical representations. |
| Data Preparation | | eccro: SFO_0000007 | functions to refine data for use in data analysis or other purposes. E.g. cleaning, QA/QC, transformations, documentation |
| Data Preparation | QA/QC | eccro: SFO_0000036 | Functions that prospectively ensure the quality of data to be acquired or which verify the data's reasonableness after acquisition through the use of techniques such as flagging outlying values, etc. |
| | Subsetting | eccro: SFO_0000039 | Functions that allow users to extract a portion of a larger data set. |

| Category | Subcategory | URI | Definition |
|---|---|---|---|
| | Metadata Assistance | eccro: SFO_0000041 | Functions that help agents develop and/or validate the metadata and other markup needed to 1) make their data usable in the short and long term or 2) prepare it for further processing and analysis. |
| | Data Cleaning | eccro: SFO_0000042 | Functions that help ensure that the data being used is clean and consistent. |
| | Data Transformation | eccro: SFO_0000310 | Functions that transform the representation of the data. |
| | Data Desensitization | eccro: SFO_0000043 | Functions that allow users to anonymize, deidentify, blur or otherwise alter or hide portions of data to protect information because of ethical and legal issues |
| Data Processing / Modeling | | eccro: SFO_0000008 | functions to produce a higher level defined data product |
| Data Processing / Modeling | Grid or Mesh Generation | eccro: SFO_0000044 | Functions that generate geometrical shapes that cover a physical domain. |
| | Test Data Generation | eccro: SFO_0000045 | Functions that generate test data. |
| | Output Validation | eccro: SFO_0000046 | Functions used to validate the output products of processing or modeling softare. |
| | Model Coupling | eccro: SFO_0000047 | Functions that allow the outputs of one model to feed into another model. |
| | Data Processing | eccro: SFO_0000048 | Functions that convert data into a useable and desired form. |
| | Computational Model | eccro: SFO_0000049 | Functions that allow users to computationally model physical processes. |
| | Model Calibration | eccro: SFO_0000050 | Functions that allow users to adjust model parameters and forcings so that model outputs match observed values. |
| | Inversion / Data assimilation | eccro: SFO_0000051 | Functions that optimally combine theory, often in the form of models, with available data. |
| | Data Integration | eccro: SFO_0000053 | Functions that combine data from different sources into a single, unified view. |
| Workflow Management | | eccro: SFO_0000009 | functions to orchestrate a series of tools in an automated fashion to perform some reproducible task. |
| Workflow Management | Reproducibility Support | eccro: SFO_0000100 | Functions that enable a workflow process to be reproduced. |

| Category | Subcategory | URI | Definition |
|---|---|---|---|
| | Provenance Management | eccro: SFO_0000101 | Functions that enable management of the provenance of 1) the definition of a workflow, 2) a workflow run, and 3) the data used or created by the workflow. |
| | Workflow Design | eccro: SFO_0000102 | Functions that enable the design or development of an orchestrated and repeatable pattern of activity. |
| | Workflow Orchestration | eccro: SFO_0000103 | Functions that enable arrangement, coordination, and management of a complex series of processes that depend on one another for their proper functioning. |
| | Workflow Execution | eccro: SFO_0000104 | |
| Data Analysis | | eccro: SFO_0000010 | functions to enable hypothesis testing or new insights from data. |
| | TIme Series Analysis | eccro: SFO_0000054 | Functions that use statistical methods to analyze time series data and extract meaningful statistics and characteristics about the data. |
| | Spatial Analysis | eccro: SFO_0000055 | Functions that use formal techniques to study entities using their topological, geometric, or geographic properties |
| | Statistical Analysis | eccro: SFO_0000056 | Functions for collecting, exploring and presenting large amounts of data to discover underlying patterns and trends. |
| | Image Processing | eccro: SFO_0000057 | Functions that allow quantitative analyses and/or algorithms applied to digital image (including multimedia) data. |
| | Text Analysis | eccro: SFO_0000058 | Functions that parse texts in order to extract machine-readable facts from them, thereby creating structured data out of free text content. |
| | Graph/Network Analysis | eccro: SFO_0000059 | Structured techniques used to mathematically analyze a network of interconnected components. |
| | Machine Learning | eccro: SFO_0000060 | Functions (i.e., algorithms and statistical models) used to perform a specific task without using explicit instructions, relying on patterns and inference observed in input test data instead. |
| Visualization | | eccro: SFO_0000011 | interaction with data to use human visual capabilities to understand the data better or communicate information from the data |
| | Display | eccro: SFO_0000062 | Functions that present data and information in a form suitable for visual perception and |

| Category | Subcategory | URI | Definition |
|---|---|---|---|
| | | | decision-making by a person. |
| | Animation | eccro: SFO_0000063 | Functions used to manipulate images or data so they appear as moving images. |
| | Rendering | eccro: SFO_0000064 | Functions used to generate two-dimensional or three-dimensional images from a model. |
| Reporting | | eccro: SFO_0000012 | functions to describe key aspects of a resource or its usage, or of new insights from data |
| | Document generation | eccro: SFO_0000065 | Functions that automate or partly automate the generation of documentation. |
| | Report Rendering | eccro: SFO_0000066 | Functions that transforms report data and layout information into a specific format. |
| | Data Reporting | eccro: SFO_0000067 | Functions to collect and submit data so that accurate analyses of the facts on the ground can be performed. |
| | Usage Monitoring | eccro: SFO_0000068 | Functions that allow the usage of a resource to be monitored. |
| Data Preservation | | eccro: SFO_0000013 | functions to maintain and conserve data in a usable and useful condition for the long term |
| | Long-term Archiving | eccro: SFO_0000105 | Functions that enable data to remain accessible and usable for a period of time long enough for there to be concern about the impacts of changing technologies, including support for new media and data formats, and of a changing Designated Community. |
| | Backup | eccro: SFO_0000106 | Functions that enable the making copies of data or data files to use in the event the original data or data files are lost or destroyed. |
| | Data Migration | eccro: SFO_0000107 | Functions that enable the preservation of digital information, through transformation of the full information content from one form to another. |
| Programmer Assistance | | eccro: SFO_0000014 | functions to support all aspects of software developement |
| | Version Control | eccro: SFO_0000071 | Functions that track changes made to a digital asset over time |
| | Code Optimization | eccro: SFO_0000072 | Functions to improve the efficiency and quality of code. |
| | Code Validation & Testing | eccro: SFO_0000073 | Functions for evaluating and ascertaining the completeness and quality of code, determining if it complies with its requirements, performs functions |

| Category | Subcategory | URI | Definition |
|---|---|---|---|
| | | | for which it is intended and meets the organization's goals and user needs. |

## ECRR controlled vocabularies

ECRR vocabularies are published via the ESIP Community Ontology Repository (http://cor.esipfed.org/ont/#/). For maintenance of vocabularies and suggestions for new terms, a listing is available in a google document (https://docs.google.com/spreadsheets/d/1ykxgdeDjBzcTc1y64CuyaS2PhDuUtLuO5AFUehwhxDs )

## EarthCube specific properties

The resource registry information model (http://cor.esipfed.org/ont/earthcube/ecrro) defines a number of properties that are not included in the Schema.org vocabulary. In some cases, properties from other vocabularies are used (e.g. dcat, Dublin Core Terms). If no good match was found in an existing vocabularies, the ECRR ontology defines a new property. In either case, our schema.org JSON-LD implementation implements these in a schema:additionalProperty array of schema:PropertyValue instances. This allows use of the schema:propertyID to contain the URI for the externally defined property, a user-friendly label for the property in the name element, and property values to be specified using various schema.org classes. Schema:PropertyValue/value elements have an expected range that includes Boolean, Number, StructuredValue, or Text. The ECRR implementation uses other schema.org classes as values, including Person, Organization, CreativeWork, and DefinedTerm. This results in errors from the schema.org validator, but the JSON-LD is valid because schema.org does not define rigid domain and range constraints on properties. Some examples:

Additional property value is a string:

```
{"@type": "PropertyValue",
 "propertyID": "dc:BibliographicCitation",
 "name": "Bibliographic citation",
 "value": {"@type": "Text",
    "value": "Martin Isenburg, 2021, LAStools, rapidlasso GmbH,
Gilching, GERMANY, https://rapidlasso.com/lastools/"
        },
```

Additional property value has a name/label and identifier, possibly a description; implement with schema:DefinedTerm.

```
{"@type": "PropertyValue",
 "propertyID": "eccro: ECRRO_0000138",
 "name": "has maturity state",
 "value": {
    "@type": "DefinedTerm",
```

```
        "name": "In production",
        "identifier": "http://cor.esipfed.org/ont/earthcube/MTU_0000002"
                }       },
```

## Potential Action

Machine actionable documentation for interactions with software agents on the web using schema.org has been a long topic of discussion in the community, e.g. [WebAPI update](https://github.com/schemaorg/schemaorg/pull/2635). Any schema.org thing has a potentialAction property with expected value of type Action or one of its subtypes. In order to implement more frictionless linkage between data and applications using the data, the GeoCODEs team has developed a set of conventions for using Action properties to support automation. The potentialAction property is used on resources that have ECRR type (schema:mainEntity) Catalog/Registry (ecrro:ECRRO_0000212), Service Instance (ecrro:ECRRO_0000202), or Software (ecrro:ECRRO_0000206).
The Actions of interest for the EarthScience research community are the functions identified in the EarthCube software function vocabulary (http://cor.esipfed.org/ont/earthcube/sfo). The ECRR action type can be categorized using the schema:additionalType property, with a range defined by the values in the EarthCube software function vocabulary (http://cor.esipfed.org/ont/earthcube/sfo).
This section starts with a summary of the properties on schema:Action, followed by the conventions for using those properties in the ECRR.

### Action properties

Result

The 'result' property documents the outcome of an Action. For the purposes of machine-actionable automation, the results of interest are file-based resources that can be delivered electronically. The purpose of the file content in a research workflow is out of scope, but documentation of the result needs to specify the kind of file produced-- the interchange format.

Target

The 'target' property documents the endpoint that receives requests to invoke an action, and the syntax for how that request is formulated.

Query-input

The 'query-input' property is a set of property values specifications that document parameters required to invoke the described action, typically slots in a urlTemplate for the target entry point.

Object

In a resource-oriented architecture on the web, requests for actions are targeted to a particular resource (the 'object'). This target resource has some electronic representation, and in some cases it is useful to know the information model for the object of an action request.

## GeoCODES Action conventions

- Action>additionalType: categorizes the action using the ECRR software function vocabulary (http://cor.esipfed.org/ont/earthcube/sfo).

Action>result

- @type: categorizes the kind of result produced by the action, using the schema.org entity types. Default value is 'schema:Dataset', for actions that return a serialized bundle of data.
- encodingFormat: an array of file-format strings, can include base MIME types, but should include types from the ECRR formats registry (https://github.com/earthcube/GeoCODES-Metadata/blob/main/resources/encodingFormat.csv ), which provide more granular categories for file types.
- conformsTo: array of identifier for specification(s) that documents the action response serialization and information model.

Action>target

- @type: MUST be schema:EntryPoint, which defines the location designation and syntax for requests to invoke the Action.
- urlTemplate: (required) For Catalog/Registry and ServiceInstance, a URL template conforming to IETF RFC6570, used to invoke the Action (parameters are documented in the following query-input element. For software, any kind of template that will assist users to invoke the Action on the software is recommended.
- httpMethod: Default is GET. Conventions for interoperable description of requests using other HTTP methods (e.g. POST, DELETE, PUT) have not been developed.
- contentType:  an array of content (MIME) types that can be requested in the HTTP accept header, if the content type is not specified in the urlTemplate. MUST be consistent with result>encodingFormat values.

Action>query-input

- An array of schema:PropertyValueSpecification that define the parameters in the target>urlTemplate.
    - valueName: (required) name of the parameter as it appears in the target>urlTemplate
    - description: (required) text explaining the usage of the parameter
    - Other properties that can be specified are documented in the schema.org PropertyValueSpecification page (https://schema.org/PropertyValueSpecification)

Action>Object

- @Type: default is DataSet, assuming the Action is invoked against some data schema. Semantics and utilization are fuzzy.
- additionalType: categorization of the kind of resource that is the object of the Action, if there is something more specific (and useful) than Dataset or one of the other Schema.org entities.

- variableMeasured: if a parameter in the URL template allows specification of one of the variables from the object data, the list of variable in the object data schema should be listed and documented here.
  Example:

```
"potentialAction": [
  {
    "@type": "SearchAction",
    "additionalType": "ecrro:SFO_0000005",
    "name": "Query",
    "description": "query service to obtain records of seismic
events",
    "result":
      {
      "@type": "DataDownload",
      "encodingFormat": [
             "application/xml+QuakeML",
             "text/csv;type=GeoCSV-GeoWS”],
      "description": "XML (QuakeML) or csv format for seismic event
            following EarthCube geoWs conventions."
      },
    "target": {
      "@type": "EntryPoint",
      "urlTemplate":"http://service.iris.edu/fdsnws/event/1/query?
                  {start}&{end}&{minmag}&{maxmag}&{format}",
      "description": "URL with multiple query parameters",
      "httpMethod":"GET",
      },
    "query-input": [
      {
      "@id": "urn:iris:fsdn.starttime",
      "@type": "PropertyValueSpecification",
      "valueName": "start",
      "description": "allowed: Any valid time. Limit to events on or
after the specified time; use UTC for time zone",
      "valueRequired": true,
      "xsd:type": "dateTime"
           },
           ... (a couple example parameters),
           ...,
            {
      "@id": "urn:iris:fsdn.maxmagnitude",
      "@type": "PropertyValueSpecification",
      "valueName": "maxmag",
      "defaultValue": "Any",
       "description": "Limit to events with a magnitude smaller than
             the specified maximum.",
      "valueRequired": true,
```

```
        "xsd:type": "float",
              }
        ]
      "object": {
        "@type": "DataSet",
        "description": "list of properties  that are included in
seismic event description in the source data system",
        "variableMeasured": [
           {
           "@type": "PropertyValue",
           "name": "name of the variable",
           "propertyID": "URI for the property in some ontology",
           "measurementTechnique": "URI for the measurement protocol,
or text description of procedure and sensor"
              }  ]  }
}
```

## Known errors

This is a list of errors that will be noted by the schema.org validator tool (https://validator.schema.org/), but are to be expected because of the design choices in the ECRR JSON-LD implementation.

*Thing* is not a known valid target type for the *contentType* property.
The property *contentType* is not recognized by the schema (e.g. schema.org) for an object of type *CreativeWork*.

In additionalProperty:
*DefinedTerm* is not a known valid target type for the *value* property.
*Organization* is not a known valid target type for the *value* property.

In registration metadata (ecrro:ECRRO_0001301):
The property *datePublished* is not recognized by the schema (e.g. schema.org) for an object of type *StructuredValue*.
The property *datePublished* is not recognized by the schema (e.g. schema.org) for an object of type *StructuredValue*.)