Notes on JSON-LD implementation for documenting Datasets for EarthCube GeoCODEs platform

Stephen M. Richard
EarthCube GeoCODES team
July 4, 2022
Current version: Google Doc

Introduction

This document provides guidance for implementing Schema.org JSON-LD metadata to document geoscience datasets that will be most compatible with the [EarthCube GeoCODES resource registry](https://geocodes.earthcube.org/). This metadata profile is based on other schema.org recommendations:

- Google Developers Guidance helps guide use in Google Dataset Search
- ESIP Science on Schema.org (SOSO) guidance for datasets

A JSON (v7) schema that specifies the JSON serialization that will work with the GeoCODES data entry forms is available on GitHub.

The following are either important in the search UI or items GeoCODES is using for value-added functionality (e.g. dataset-tool linkage). It is important to know that these are items used by the search application. This profile is based on the ESIP Science on Schema.org guidance, with some additional restrictions and recommendations.

Mandatory Properties

Automatically populated

The following fields should be automatically populated by any metadata creation software.

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.

JSON-LD Object Type

- JSON key: '@type'.
- Value: Class name from schema.org vocabulary; For data sets, the value MUST be 'Dataset'.

Type of Resource

- JSON key: mainEntity
- This will be a constant for Datasets. The expected value is:

```
"mainEntity":[{
        "@type": "CreativeWork",
        "url": "http://schema.org/Dataset",
        "name": "Dataset"
        }  ],
```

User provided

Resource Name

- JSON key: 'name'
- Value: text
- Short name by which this Dataset will be recognized by human users; 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 Dataset. This text will be indexed by search aggregators, and the information contained should be sufficient to tell a person what is in the dataset, how it was acquired, any processing, and broadly how to use it. 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 data. Note it can 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

License Name	URI
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"
        }  ],
```

Is available for free

- JSON key: 'isAccessibleForFree'
- Boolean value (true or false). True if the dataset is available at no (monetary) cost to the user.

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 formal metadata record

- JSON key: subjectOf
- Value: Array of labeled link objects;
- The URL in the labeled link should access a metadata record using one of the more formal (and complete) metadata schemes, like ISO19115 or EML.

Example:

```
 "subjectOf": [
   {"@type": "CreativeWork",
    "name": "ISO 19115 metadata",
    "url":"http://ds.iris.edu/files/metadata/19115/376577"
   }  ],
```

Preferred Dataset Identifier

- JSON key: identifier
- Value: string
- A globally-unique URI, ideally using a scheme that can be dereferenced on the Web. HTTP URI is preferred, but any URI scheme is allowed, e.g. ISSN, DOI, ARK, ISBN, URN:OGC.. For some formats, a legacy identifier string is in use and is also included.

Example.

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

Other identifiers

- JSON key: sameAs
- Value: array of strings
- Other identifier strings that have been used to identify the dataset. These might be URIs or local identifiers.

Example:

```
"sameAs": ["http://repository.azgs.az.gov/uri_gin/azgs/dlio/245"],
```

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"
        }
    ],
```

Registration metadata.

This is a GeoCODES addition to track the provenance of the metadata content.

- 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"
    }
   }
```

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"].
  ```

Version of the Dataset

- JSON key: version
- Value: string
- This property identifies a particular version of the Dataset 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 dataset. Note that the current (2022-02) 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.

Legacy 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"
            }
        }  ],
```

Current (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 (Maintainer) 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 (schema:maintainer is proposed in schema.org, and will replace this property when adopted). 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/"}
      },
```

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:Product or 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.

Distribution

The distribution elements in Dataset metadata records describe the options of acquiring the actual data, The schema data type for the distribution content is necessarily ‘schema: DataDownload’. The distribution patter follows recommendations in the ESIP Science on Schema.org recommendations.
While the schema:url property of the Dataset should point to a landing page (web site), the schema:distribution property describes how to download the data in various formats or through different WebAPIs. The "distribution" property describes links on the web to get the data, and the formats offered at those endpoints using the schema:DataDownload type. If your dataset is not accessible through a direct download URL, the DataDownload link provided must be either schema:url linking to a dataset landing page or form, or typed as a ‘WebAPI’ in addition to DataDownload to document access through a service URL that may need input parameters jump to the next section Accessing Data through a Service Endpoint.

Spatial Extent

Spatial Extent document in GeoCODEs follows Science on Schema.org recommendations, with the limitation that extents can only be expressed as a single point, a collection of points, a bounding box, or a set of bounding boxes.
The Spatial Extent object documents the location on Earth that is the focus of the dataset content, using schema:Place. For GeoCODES compatibility, use the schema:geo property with either a schema:GeoCoordinates object to specify a point location, or a schema:GeoShape/schema:box object to specify an area coverage extent. Coordinates describing these extents are expressed as latitude longitude tuples (in that order) using decimal degrees.

Schema.org documentation does not specify a convention for the coordinate reference system; GeoCODES assumes the reference system is WGS84. Spatial coverage location using other coordinate systems can be included, see recommendation for specifying coordinate reference systems, below.

Use GeoCoordinates for Point locations

Please indicate a point location by using a schema:GeoCoordinates object with schema:latitude and schema:longitude properties.

Point locations are recommended for data that is associated with specific sample locations, particularly if these are widely spaced such that an enclosing bounding box would be a misleading representation of the spatial location. Be aware that some client applications might only index or display bounding box extents or a single point location.

A schema:Dataset that is about a point location would documented in this way:

```
{
  ...
  "spatialCoverage": {
    "@type": "Place",
    "geo": {
      "@type": "GeoCoordinates",
      "latitude": 39.3280
      "longitude": 120.1633
    }
  }
}
```

Use bounding box for area extents

A GeoShape box defines a rectangular area on the surface of the Earth defined by point locations for the southwest corner and northeast corner of the rectangle in latitude-longitude coordinates. Point locations are tuples of {latitude east-longitude} (y x). The schema.org GeoShape documentation states "*Either whitespace or commas can be used to separate latitude and longitude; whitespace should be used when writing a list of several such points*." Since the box is a list of points, a space should be used to separate the latitude and longitude values. The two corner coordinate points are separated by a space. 'East longitude' means positive longitude values are east of the prime (Greenwich) meridian. A box where 'lower-left' (southwest) corner is 39.3280/120.1633 and 'upper-right' (northeast) corner is 40.445/123.7878 would be encoded thus:

```
 "box": "39.3280 120.1633 40.445 123.7878"
```

For GeoCODES, East longitude values should be reported -180 <= X <= 180, consistent with the WKT specification. Following this recommendation, bounding boxes that cross the antimeridian at ±180° longitude, the West longitude value will be numerically greater than the East longitude value. For example, to describe Fiji the box might be

```
 "box": "-19 176 -15 -178"
```

NOTES: Some spatial data processors will not correctly interpret the bounding coordinates across the antimeridian even if they follow the recommended southwest, northeast corner convention, resulting in boxes that span the circumference of the Earth, excluding the actual area of interest. For applications operating with data in the

vicinity of longitude 180, testing is strongly recommended to determine if it works for bounding boxes crossing the antimeridian (+/- 180); an alternative is to define two bounding boxes, one on each side of 180.

For bounding boxes that include the north or south pole, schema:box will not work. Recommended practice is to use a schema:polygon to describe spatial location extents that include the poles.

Handling multiple locations

If the spatial extent is best represented with multiple geometries, can publish those by making the schema:geo field an array of GeoShape or GeoCoordinates like so:

```
{
  ...
  "spatialCoverage": {
    "@type": "Place",
    "geo": [
      {
        "@type": "GeoCoordinates",
        "latitude": -17.65,
        "longitude": 50
      },
      {
        "@type": "GeoCoordinates",
        "latitude": -19,
        "longitude": 51
      },
      ...
    ]
  }
  ...
}
```

Be aware that some client application might not index or display multiple geometries.

Temporal Extent

Temporal coverage is defined as "the time period during which data was collected or observations were made; or a time period that an activity or collection is linked to intellectually or thematically (for example, 1997 to 1998; the 18th century)" (ARDC RIF-CS). For documentation of Earth Science, Paleobiology or Paleontology datasets, we are interested in the second case-- the time period that data are linked to thematically.

Temporal coverage is a difficult concept to cover across all the possible scenarios. Schema.org uses ISO 8601 time interval format to describe time intervals and time

points, but doesn't provide capabilities for geologic time scales or dynamically generated data up to present time.

Geologic Time
Dates or ages used for describing geological, archeological, and paleontological samples range from the very simple to highly complex. A lava rock age could be simply described as 1.23 million years. Other ages are more descriptive. Some other examples are: a zircon crystal with an age of 456.4 +/- 1.4 billion years (Ga) at a standard error of 2-sigma, a core with rocks from the Triassic to the Jurassic, a carbon date of a bone with non-symmetrical uncertainties of 3242 (+160 -40) B.C. We make use of the OWL time (Cox and Little) descriptive tags (elements), the Queensland Department of Natural Resources, Mines and Energy Temporal Reference Systems (TRS), and geoschemas' properties to describe ages and age ranges in detail. These methods could also be used to describe the temporal coverage for other disciplines as well.

There are two main types of geologic time: Proper Intervals and Instants. They are diagrammed below and used in the examples that follow.

Variable Measured

A Dataset is a collection of data entities, each of which contains structured and unstructured values for a set of properties about that entity. For example, consider three kinds of datasets that might be collected about lakes: 1) a data table in CSV format containing columns of data that both classify and measure the properties of a set of lakes in a region; 2) an image file containing rasterized geospatial data values for each location for properties like water temperature at multiple depths; and, 3) a text file containing responses to a survey assessing perspectives on water rights, with values for questions containing both natural language responses and responses on a Likert scale. In each of these examples, we are recording the value of attributes (aka properties) about an entity of interest (lake). In schema.org, details about these attributes can be recorded using `schema:variableMeasured`. So, while schema.org uses the term "variable" and the term "measured", it is conceptualized as a listing of any of the properties or attributes of an entity that are recorded. Use `schema:variableMeasured` to represent any recordable property of an entity that is found in the dataset. While this includes quantitatively "measured" observations (e.g., rainfaill in mm), it also includes classification values that are asserted or qualitatively assigned (e.g., "moderate velocity"), contextual attributes such as spatial locations, times, or sampling information associated with a value, and textual values such as narrative text.

Schema.org allows the value of `schema:variableMeasured` to be a simple text string, but it is strongly recommended to use the `schema:PropertyValue` type to describe the variable

in more detail. The ESIP Science on Schema.org recommendations outline several tiers of variable description. Tier 1 is the simplest, with other tiers adding recommendations for additional content (Tier 2 and 3). An Experimental Recommendations document contains proposed recommendations for variables with with non-numeric or enumerated (controlled vocabulary) values, variables whose values are structured objects (e.g. json objects, arrays, gridded data), or are references to external value representations. The SOSO tiers are as follows:

Tier 1. Simple list of variable names

Provide a `schema:name` and a textual description of the variable. The `schema:name` should match the label associated with the variable in the dataset serialization (e.g. the column name in a CSV file). If the variable name in the dataset does not clearly convey the variable concept, a more human-intelligible name can be provide using `schema:alternateName`. The field `schema:description` is used to provide a definition of the variable/property/attribute that allows others to correctly understand and interpret the values.

Example:

```
{
  "@context": "https://schema.org/",
  "@type": "Dataset",
  "name": "Removal of organic carbon by natural bacterioplankton communities ...",
  ...
  "variableMeasured": [
    {
      "@type": "PropertyValue",
      "name": "latdd",
      "alternateName":"latitude, decimal degrees",
      "description": "Latitude where water samples were collected ...",
    },
    ...
  ]
}
```

Tier 2: Names of variables with formal property types

Use `schema:PropertyValue` object to provide a `schema:propertyID` that better defines the semantics of the variable than plain text can. This `schema:propertyID` should be a URI that resolves to a web page providing a human-friendly description of the variable and, ideally, this identifier should also be resolvable to obtain an RDF representation using a documented vocabulary for machine consumption, for example a sosa:Observation or DDI represented variable. Describing the variables with machine understandable vocabularies is necessary if you want your data to be interoperable with

other data, i.e., to be more FAIR. The property can be identified at any level of specificity, depending on what the data provider can determine about the interpretation of the variable. For example, one might use a propertyID for the property 'temperature', or use a more specific property like 'water temperature', 'sea surface water temperature', or 'sea surface water temperature measured with protocol X, daily average, Kelvins, xsd:decimal'. If there are choices, the most specific property identifier should be used.

Example:

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@type": "Dataset",
  "name": "Removal of organic carbon by natural bacterioplankton communities ...",
  ...
  "variableMeasured": [
    {
      "@type": "PropertyValue",
      "name": "latdd",
      "alternateName":"latitude, decimal degrees",
      "propertyID":"http://purl.obolibrary.org/obo/NCIT_C68642",
      "description": "Latitude where water samples were collected ...",
    },
    ...
  ]
}
```

Tier 3: Numeric values

For variables with numeric measured values, other properties of schema:PropertyValue can add additional useful information:

- schema:unitText. A string that identifies a unit of measurement that applies to all values for this variable.
- schema:unitCode. Value is expected to be TEXT or URL. We recommend providing an HTTP URI that identifies a unit of measure from a vocabulary accessible on the web. The QUDT unit vocabulary provides an extensive set of registered units of measure that can be used to populate the schema:unitCode property to specify the units of measure used to report data values when that is appropriate.
- schema:minValue. If the value for the variable is numeric, this is the minimum value that occurs in the dataset. Not useful for other value types.

- schema:maxValue. If the value for the variable is numeric, this is the maximum value that occurs in the dataset. Not useful for other value types.
- schema:measurementTechnique. A text description of the measurement method used to determine values for this variable. If standard measurement protocols are defined and registered, these can be identified via http URI's.
- schema:url Any schema:Thing can have a URL property, but because the value is simply a url the relationship of the linked resource can not be expressed. Usage is optional. The recommendation is that `schema:url` should link to a web page that would be useful for a person to interpret the variable, but is not intended to be machine-actionable.

Example:

```
{
  "@context": {
    "@vocab": "https://schema.org/"
  },
  "@type": "Dataset",
  "name": "Removal of organic carbon by natural bacterioplankton communities as a
function of pCO2 from laboratory experiments between 2012 and 2016",

  "variableMeasured": [
    {
      "@type": "PropertyValue",
      "name": "latitude",
      "propertyID":"http://purl.obolibrary.org/obo/NCIT_C68642",
      "url": "https://www.sample-data-repository.org/dataset-parameter/665787",
      "description": "Latitude where water samples were collected; north is positive.
Latitude is a geographic coordinate which refers to the angle from a point on the
Earth's surface to the equatorial plane",
      "unitText": "decimal degrees",
      "unitCode":"http://qudt.org/vocab/unit/DEG",
      "minValue": "45.0",
      "maxValue": "15.0"
    },
    ...
  ]
}
```

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/"
    } ],
```

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**.

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