Having your cake and eating it too: JSON-LD as an RDF serialization format
https://doi.org/10.3897/biss.5.74266
Steven J. Baskauf https://orcid.org/0000-0003-4365-3135
2021-10-18
TDWG Conference 2021

Draft Design Patterns for use of JSON-LD across TDWG

1 Overview

This document is modeled after the IIIF Design Patterns document, https://iiif.io/api/annex/notes/design_patterns/ . It draws mainly from the sections of the IIIF document relating to use of JSON-LD (Sections 2.7, 2.8, and some parts of Section 3), although other parts may more broadly relevant. This document also considers the W3C's JSON-LD Best Practices Working Group Note: https://w3c.github.io/json-ld-bp/ . However, this document diverges from the recommendations of these other groups in cases where the needs and past practices of TDWG argue for a different approach.

The specific recommendations are grouped within three overarching principles that guided their development.

1.1 RFC 2119 key words

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

1.2 JSON terminology

When discussing the structure of JSON, the words "object", "array", "name", and "value" have the meanings given by the JSON specification at https://www.json.org/ .

2 The "Simple Darwin Core" principle

As the first "modern" ratified TDWG vocabulary, Darwin Core (DwC) set a number of precedents that were followed by Audubon Core (AC), and which were in some cases codified in the Standards Documentation Specification (SDS). One of these precedents was that in its simplest form, data can be transmitted in tabular form, where columns represent properties and rows represent instances of the type of resource described by the table. As described in the Simple Darwin Core guide, column headers denote the property "term names" (i.e. abbreviated IRIs or CURIEs), while the cells contain the values of the column property for the resource described by the row. For simplicity, deep nesting of the sort often found in XML schemas was avoided.

This style was followed for descriptions of TDWG vocabulary terms themselves when the SDS stipulated that the basic layer for vocabularies should consist of a "bag of terms", each described

by a few key properties that can be expressed in "flat" tables similar to Simple Darwin Core tables. All basic metadata about TDWG vocabularies is expressed in such tables in the authoritative rs.tdwg.org GitHub repository from which all vocabulary documents and machine-readable metadata are generated.

Since data to be serialized as JSON-LD is likely to originate from tables like these, the JSON should be structured in a manner that imitates the table organization, making it relatively simple to map the tabular data to JSON.

2.1 @graph array

The main objects described MUST be values in an array that is a value of a top-level "@graph" name. (The @graph array corresponds to a data table and each object in the array corresponds to a row in the table.)

2.2 names in object descriptions

The names of name/value pairs describing objects MUST be CURIEs (a.k.a. abbreviated IRIs or "term names" sensu SDS section 3.3.3.1). The namespace abbreviations MUST be defined in the @context section of the document. (Note: this differs significantly from the IIIF recommendations. Whereas names in IIIF JSON-LD documents typically come from only a few well-known namespaces, TDWG vocabularies use many namespaces, which often have terms with identical local names. For example, it must be possible to distinguish between dc:type and dcterms:type, or dwc:recordedBy and dwciri:recordedBy.)

2.3 values in object descriptions

Whenever possible, the values in object descriptions SHOULD mirror the forms they would take in a table. For example, native JSON datatypes should be used for numbers (JSON-LD Best Practice 3). Generally, this means avoiding nested objects and arrays as values whenever there is an alternative method that makes this possible (see Section 4 below).

2.4 IRIs identifying objects

When IRIs are used to identify objects, a fully expanded IRI MUST be used. This is to allow values to be taken directly from table cells, where namespace abbreviations would be unspecified (making CURIE values ambiguous).

3 The "star schema" principle

The Darwin Core Text Guide describes a system where one or more extension tables can be linked to a core table, essentially enabling description of a graph limited to nodes located no further than a single edge from a core resource. Similarly, the Audubon Core Structure document describes ways that tabular data can link multiple ServiceAccessPoint instances to media items. In both of these systems, it is not required that the many "extension" resources be denoted by

IRIs. Thus, it is necessary that these design patterns support at least one level of nesting to allow for links to blank nodes.

JSON-LD Best Practice 9 suggests that "when multiple related entity descriptions are provided inline, related entities SHOULD be nested." This recommendation is somewhat at odds with the goal to avoid deep nesting expressed in the "Simple Darwin Core" principle. However, allowing a single level of nesting facilitates the linking of unidentified closely related resources such as the use cases mentioned in the previous paragraph.

3.1 The level of nesting below the main described objects SHOULD be limited to one.

3.2 If a property can ever have multiple values, the value MUST be an array even if there is only a single value in a particular instance (see IIIF Design pattern 3.4.1).

3.3 The @type of the main described object SHOULD be identified (JSON-LD Best Practice 6). The type of nested objects MAY be identified, although the linking property may make the type obvious (for example ac:hasServiceAccessPoint would only link to ac:ServiceAccessPoint instances). [Need to consider this in light of IIIF Design pattern 3.4.2]

3.4 The @id of the main described object SHOULD be given whenever an IRI has been minted for that object. This allows the object to be linked to resources external to the document. The @id of nested objects SHOULD be given whenever they do not represent blank nodes. Again, this allows them to be linked to external resources. For example, a key use case for Regions of Interest in media items is to allow links to be made to specific features of a media item. Thus IRIs for them should be minted and exposed even though the main described item would be the entire media item.

4 The "simple but self-describing" principle

There is somewhat of a tradeoff between describing values fully and keeping the structure of the JSON simple. In many cases, the solution to this problem is to define global typing in the @context.

4.1 Properties expecting IRI values MUST be so designated through a {"@type": "@id"} value in the @context. For example, this should be the case for all Darwin Core dwciri: properties and all Audubon Core properties that have xLiteral analogs. Making this designation in the @context allows the data values to be expressed as simple string values that are fully expanded IRIs, just as they would be given in a table. Designating links to external resources as IRI values alerts applications to the possibility that more information about those resources may be discoverable, or that the link may lead to additional information if there is a union of the JSON-LD graph with another graph where the IRI is the subject of additional RDF triples.

4.2 Whenever possible, datatyped values SHOULD have the datatype designated for the property in the @context; for example xmp:CreateDate: {"@type": "http://www.w3.org/2001/XMLSchema#dateTime"} . This might not be possible if data are inconsistently typed for a particular property.

4.3 The @context SHOULD be included within the JSON-LD document rather than through reference to an external context. This makes the document completely self-describing and eliminates the need for a consuming application to dereference a URL reference to the external context. In the future, if communities of practice develop consensus contexts, it would be more practical for applications to cache relatively few contexts and simplify the JSON-LD documents by eliminating the need for an explicit @context section within the document.