PROV-JSONLD: A JSON and Linked Data Representation for Provenance

Trung Dong Huynh, Danius T. Michaelides, and Luc Moreau

Electronics and Computer Science, University of Southampton
Southampton, SO17 1BJ, United Kingdom
{tdh,dtm,l.moreau}@ecs.soton.ac.uk

Abstract. In this paper, we propose a representation for PROV in JSON-LD, the JSON format for Linked Data, called PROV-JSONLD. As a JSON-based format, this provenance representation can be readily consumed by Web applications currently supporting JSON. As a Linked Data format, at the same time, it also represents provenance data in RDF using the PROV ontology. Hence, it is suitable for usages in both the Web and the Semantic Web.

1 Introduction

PROV provenance currently can be serialised in a number of representations: PROV-N, PROV-XML, PROV-JSON, or in a RDF serialisation using the PROV Ontology (PROV-O).[1] The latter, arguably, is most suitable for Linked Data, given that they can be readily consumed by existing Semantic Web tools and comes with the semantic grounding provided by PROV-O. There are, however, two main challenges for web-based applications when consuming PROV provenance represented in RDFs: (1) Many web applications are built to be light-weight, working mainly with simple data formats like JSON,[2] not the semantically rich RDF data model; (2) there are different, valid ways of representing PROV in RDF and such applications typically do not have the capability to infer canonical provenance information from such flexible representations. The JSON-LD format[3] partly addresses the first challenge by encoding RDF data in JSON. However, serialising RDF data into JSON-LD does not always produce JSON data in an expected structure, thus, necessitating the capability to understand the RDF data model to correctly interpret the data.

Against this background, we propose a new representation for PROV based on JSON-LD, called PROV-JSONLD, to address the above challenges. Specifically, PROV-JSONLD specifies a number of rules for representing PROV provenance in JSON-LD to provide a predefined data structure for PROV in JSON. As a JSON-based format, PROV-JSONLD can be easily consumed and processed

[1] See https://www.w3.org/Submission/prov-json/ for PROV-JSON and https://www.w3.org/TR/prov-overview/ for the other PROV representations.
[2] The JavaScript Object Notation (JSON): https://tools.ietf.org/html/rfc7159.
[3] JSON-LD: https://www.w3.org/TR/json-ld/.

by web applications and clients that are already supporting JSON without the need for the to understand the RDF data model. Moreover, with the emerging popularity of JSON databases, PROV-JSONLD allows provenance information to be stored along with application data as-is. As a Linked Data format, PROV-JSONLD is fully compatible with the RDF data model and PROV-O. As such, it can be readily processed by existing tools built for Linked Data and the Semantic Web. Last but not least, PROV-JSONLD was designed to serialise individual PROV records, as units of information in PROV, wholly into separate JSON objects. By so doing, the format enables the processing of large amount of provenance data in a stream-like fashion, which is not currently possible with PROV-JSON or other RDF serialisations.

2 JSON-LD representation for PROV

PROV-JSONLD is a variant of JSON-LD designed with the following key principles: (1) for every type of PROV record, there is only one way to represent it in PROV-JSONLD; (2) each PROV record is wholly contained in one JSON object; and (3) except from the active JSON-LD context, no other information outside the JSON object is required to interpret the record.

2.1 Encoding a PROV-JSONLD document

A PROV-JSONLD document is a self-contained package of PROV records in the form of a JSON-LD document. We provide the **PROV-JSONLD context** at `https://provenance.ecs.soton.ac.uk/prov.jsonld` to disambiguate terms used by PROV-JSONLD and map them to IRIs. It defines:

- `prov:` and `xsd:` prefixes (similarly predefined in PROV-N).
- The expected data types of PROV properties used in PROV-JSONLD.
- Simplified terms for all PROV-O properties used by PROV-JSONLD, e.g. `entity` for `prov:entity`, `activity` for `prov:activity`, and so on.
- Special terms introduced by PROV-JSONLD (listed in Table 1).

A PROV-JSONLD document optionally starts with the `@context` object, providing a reference to the PROV-JSONLD context above and the definition for any extra prefix. A default namespace can also be provided in the context's `@base` property. The PROV records packaged by the document go into its `@graph` array, each in a separate JSON object.

2.2 Encoding a PROV record

Each PROV record is serialised into a single JSON object in a `@graph` array. All the constituents of the PROV record become properties of the node as follows:

- The **identifier** (if present) becomes the identifier of the node (`@id` property).
- The **type** of the record, e.g. Activity, Derivation, becomes the **first** type of the node. Additional types are added to the `@type` array if they are valid IRIs; the remaining types become `prov:type` property of the node.

```
{ "@context": [
    "https://provenance.ecs.soton.ac.uk/prov.jsonld",
    { "@base": "http://example.org/",
      "foaf": "http://xmlns.com/foaf/0.1/" }
  ],
  "@graph": [ ] }
```

Listing 1: The basic structure of a PROV-JSONLD document.

```
{ "@graph": [
    { "@id": "exg:correct1", "@type": "prov:Activity",
      "startedAtTime": "2012-03-31T09:21:00.000+01:00",
      "endedAtTime": "2012-04-01T15:21:00.000+01:00"
    } ] }
```

Listing 2: An Activity record in PROV-JSONLD.

- **PROV attributes** are mapped into the corresponding PROV-O properties.
- **Additional attributes** become properties of the node.

For PROV elements, applying the above PROV-JSONLD encoding rules is straightforward. With PROV relations, however, there are some exceptions due to the multiple ways of representing them in RDF. In order to ensure a unique JSON structure for each type of PROV relation and that each relation record be encoded wholly in a single node, PROV-JSONLD only uses the qualified form of a PROV relation. Doing so, however, creates two encoding challenges.

First, although PROV-O defines classes for the qualified relations, the ontology does not specify a property to relate those qualified relations to the "subject" of the relations. Instead, for each qualified relation, it defines a qualification property to link the subject to the qualified relation in that order. In order to encode the subject in the same node as the relation, we introduce reverse properties using the `@reverse` mechanism provided by JSON-LD. Specifically, the subject of a PROV relation becomes a property of the node encoding the relation according to Table 1.

In addition, for Revision, Quotation, and PrimarySource records, which are subtypes of Derivation, PROV-JSONLD represents them as Derivation records with an additional types (`prov:Revision`, `prov:Quotation`, and `prov:PrimarySource`, respectively). This is to enable web applications, which typically do not have inference capabilities, to interpret such records as Derivation records when consuming PROV-JSONLD. Listing 3 below illustrates the encoding rules above for a Revision record. Note that this record does not have an `@id` property as it does not have an identifier; and the `entity_derived` property is a reverse property.

The second encoding challenge is that PROV-O does not define a qualified relation for Specialization, Alternate, and Membership; as a result, those records require special encoding rules. The only way to represent them in RDF is with the `prov:specializationOf`, `prov:alternateOf`, and `prov:hadMember` properties, respectively. For those records, PROV-JSONLD encodes them in single nodes by using the "subject" of such a relation as the node's identifier and the

Table 1. PROV-JSONLD terms and their (reverse) PROV-O predicates

PROV-JSONLD terms	PROV-O qualification properties	PROV record
entity_generated	prov:qualifiedGeneration	Generation
entity_derived	prov:qualifiedDerivation	Derivation
entity_invalidated	prov:qualifiedInvalidation	Invalidation
entity_attributed	prov:qualifiedAttribution	Attribution
activity_using	prov:qualifiedUsage	Usage
activity_started	prov:qualifiedStart	Start
activity_ended	prov:qualifiedEnd	End
activity_associated	prov:qualifiedAssociation	Association
informed	prov:qualifiedCommunication	Communication
delegate	prov:qualifiedDelegation	Delegation
influencee	prov:qualifiedInfluence	Influence

```
{  "@graph": [
    { "@type": ["prov:Derivation", "prov:Revision"],
      "entity_derived": "exg:dataset2",
      "entity": "exg:dataset1",
      "hadActivity": "exg:correct1"
    } ] }
```

Listing 3: A Revision record in PROV-JSONLD.

"object" as the value of the appropriate property from the three above. There is no `@type` or any additional property for those nodes as they are not allowed by PROV. For example, the PROV-N statement `alternateOf(exg:dataset2, exg:dataset1)` is represented in PROV-JSONLD as: `{ "@graph": [ { "@id": "exg:dataset2", "alternateOf": "exg:dataset1" } ] }`.

2.3 Encoding a PROV bundle

Provenance records can be bundled into a named set called a bundle. Following examples in the PROV-O specification, PROV-JSONLD uses named graphs to represent a named set of PROV records. In particular, PROV-JSONLD represents a PROV bundle similarly to the way a PROV document is encoded in Sect. 2.1. In this case, however, the `@graph` array is paired with an `@id` property, which encodes the bundle's identifier.

3 Conclusions

In this paper, we propose a new representation for PROV based on JSON-LD. Given its compatibility with both JSON and the RDF data model, in addition to being amenable to stream processing, PROV-JSONLD has a vast range of useful applications. We believe that its introduction will contribute positively to the adoption of PROV as the provenance standard of choice in Linked Data and Web applications alike.