Implementing a New FHIR RDF Specification for Semantic Clinical Data Using a JSON-LD-based Approach

Deepak K. Sharma[1], Eric Prud'hommeaux[2], David Booth[3], Kevin J. Peterson[1], Daniel J. Stone[1], Harold Solbrig[4], Guohui Xiao[5], Emily Pfaff[6], Guoqian Jiang[1]
[1]Mayo Clinic, Rochester, MN, USA; [2]Janeiro Digital, Boston, MA, USA; [3]Yosemite Project, Somerville, MA, USA; [4]Johns Hopkins University, Baltimore, MD, USA; [5]Free University of Bozen-Bolzano, Italy; [6]University of North Carolina, Chapel Hill, NC, USA

Abstract. FHIR RDF enables operational healthcare data to be linked with RDF data from other communities. FHIR data can be serialized in either JSON, XML or RDF (Turtle), and tools are used to convert between formats. However, currently the tools for converting to/from FHIR RDF involve custom code. JSON-LD 1.1 @context files now have potential to reduce the cost of implementing and maintaining this FHIR RDF conversion. These @context files can be generated automatically during the FHIR specification build process. Used with a standard, off-the-shelf JSON-LD 1.1 processor, these @context files can do most of the work needed for this conversion, though a small amount of pre- or post-processing is still needed. Using the latest FHIR build and server implementations, we created a framework for the FHIR RDF specification implementation by developing two tools to demonstrate this process: a JSON-LD 1.1 @context generator that produces @context files from the FHIR specification; and a command-line tool for batch conversion between FHIR JSON and FHIR RDF.

Keywords: HL7 FHIR; Healthcare; JSON-LD; Shape Expression; ShEx; RDF; JSON-LD Context.

1. Introduction

HL7 Fast Healthcare Interoperability Resources (FHIR)[1] is a next generation standards framework for exchanging electronic healthcare data. The Semantic Web Resource Description Framework (RDF) is one of three standard formats for FHIR data: JSON[2], XML[3] and RDF[4] (Turtle). The HL7 FHIR RDF Workgroup is actively working on creating a new version of the FHIR RDF specification, aiming to achieve better utility and ease of use. Early adopters of FHIR RDF uncovered a number of issues, including 1) literal values are nested under blank nodes (BNodes); 2) FHIR References are nested under BNodes; 3) Ordered lists use an extra BNode to include an explicit `fhir:index;` 4) FHIR RDF uses long predicate names to ensure uniqueness; and 5) FHIR extensions are awkward for RDF users. For this reason, the FHIR RDF Workgroup is now placing greater emphasis on ease of use.

To address these problems, we explored the use of JSON for Linked Data, version 1.1. (JSON-LD 1.1)[5] as a means to generate an executable definition of the FHIR RDF specification to enable semantics of the FHIR data. Note that JSON-LD is a W3C standard that allows JSON-LD data to be interpreted as a serialization of RDF triples. JSON-LD 1.1 enables the FHIR RDF specification to be defined in a more declarative style, through the use of "@context".

JSON LD @context files act like a schema for the JSON instance data and can be used to interpret and translate JSON data into RDF formats. The same instance data can be interpreted in multiple ways with different sets of JSON LD @context files. While the JSON LD 1.1 approach is an attractive approach, there are some issues uncovered by early adopters[6]. We focus on these issues here to ensure the implemented tools provide a simple and practical way to convert FHIR JSON to RDF.

FHIR JSON can be validated with JSON Schema files that are published with FHIR. However, the JSON schema language has limited expressivity. For instance, it doesn't reject mutually

exclusive properties. You can better verify conformance with the FHIR specification by converting the JSON data to RDF and validating the result with Shape Expression (ShEx)[7] tools.

In Phase I of this effort, we re-implemented the existing FHIR RDF specification (R4) using JSON-LD 1.1 and demonstrated that this was effective and more declarative than the existing custom-code-based approach. The objective in Phase II is to demonstrate the feasibility of the JSON-LD 1.1 based approach to produce executable definitions of a new FHIR RDF specification.

2. Methods:

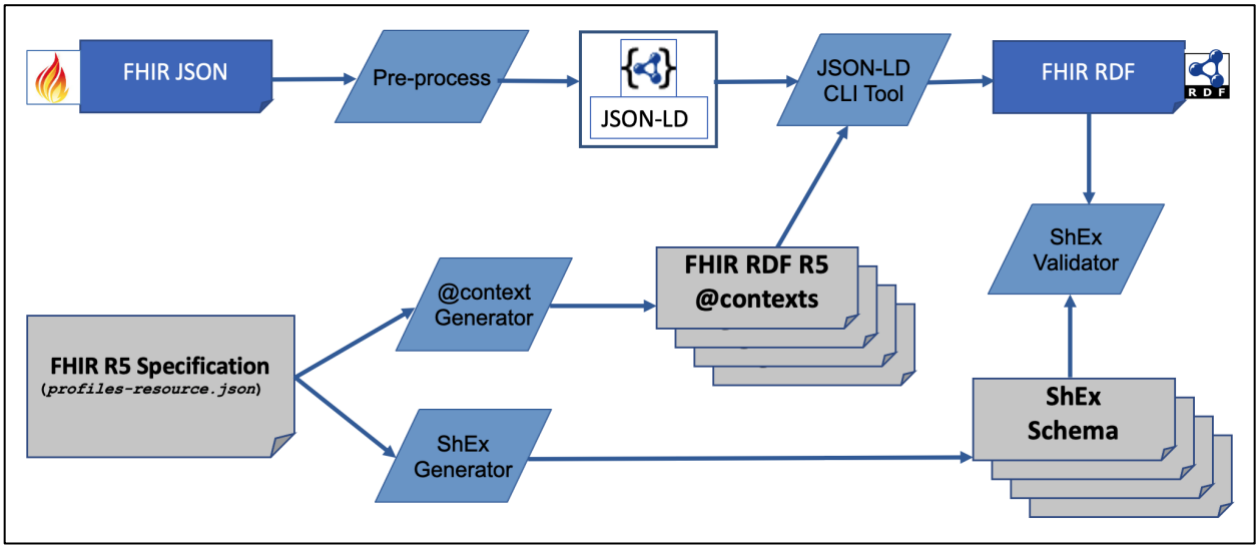

Figure 1. A JSON-LD-based framework for FHIR RDF specification implementation

2.1. Creating a draft FHIR RDF R5 specification

The initial draft version of the FHIR RDF R5 specification addresses the following issues:

- BNodes: FHIR RDF R4 included extra BNodes to express nested values, and references. For example, in FHIR RDF R4 a status of "final" is represented as `fhir:status [fhir:value "final"]`, instead of simply `fhir:status "final"`, which complicates how these values are queried by a SPARQL query. In R5 we intend to eliminate these extra BNodes.

- Shortened predicate names: the predicate names in FHIR R4 were unnecessarily long and could be shortened to reduce redundancy and simplify usage, e.g., instead of having separate `fhir:AdverseEvent.identifier` and `fhir:AllergyIntolerance.identifier` properties, a single shorter `fhir:identifier` property could be used across multiple resources.

  The FHIR model is built with resources descending from a common base class `StructureDefinition` while data types descend from base classes of `Element` or `BackboneElement`. We shortened the predicate names if they are not core `Element` or `BackboneElement`. An example of this would be predicate `fhir:observation.status` transformed to simply `fhir:status`.

- Extensions: FHIR allows extensions on Resources and on properties in Resources. Extensions are classed as "modifying" or "non-modifying"; where modifying extensions change the interpretation of the enclosing element and must not be interpreted without understanding the extension. The FHIR RDF group is balancing tensions between RDF's monotonicity vs. keeping the RDF representation like JSON and XML.

2.2. Developing a @context generator for implementing R5 specification

- JSON LD 1.1 @context Generator: We developed a Java-based JSON LD @context Generator[8] that generates JSON-LD 1.1 @contexts[9] for the new draft specification out of the `StructureDefinitions` of the FHIR data models. The FHIR `StructureDefinition` resource describes a structure, i.e., a set of data element definitions, and their associated rules of usage.

- Shape Expression (ShEx) Generator: We also extended the Shape Expression Generator[10] in the FHIR Builder to produce the ShEx schemas[11] of the FHIR data models.

2.3. Implementing a JSON-LD command line tool for data transformation

We implemented a JSON-LD 1.1 FHIR RDF Java-based command line tool[12] to convert the JSON files of the FHIR R4 and R5 examples into the RDF Turtle representation that conforms to the new FHIR RDF specification. The command line tool takes care of pre- and post-processing of FHIR JSON instance data to be properly used with the JSON-LD 1.1 @context files. The command line tool also supports optional ShEx validation to ensure converted RDF graphs conformant to the FHIR RDF specification as defined in ShEx schemas.

3. Results:

The JSON-LD 1.1 @context generator produced 1355 @context files[9] after processing 658 `StructureDefinition`s of all published FHIR artifacts[13] - including resources, foundational elements, patterns and terminologies. The JSON-LD 1.1 @context files produced correct rendering identifiers and @context files referenced to work together. An example of this is `MedicationRequest.dosageInstruction` of resource type `Dosage`.

JSON-LD 1.1 @context file `medicationrequest.context.jsonld` contains references to `Dosage` type:

```
"dosageInstruction": {
      "@id": "fhir:dosageInstruction",
      "@context": "dosage.context.jsonld"
    },
```

This `Dosage` type is defined in its own @context file `dosage.context.jsonld`, which defines Dosage. This example also demonstrates shortening of the name from `fhir:MedicationRequest.dosageInstruction` to `fhir:dosageInstruction`.

Figures 2 and 3 demonstrate transformation of FHIR JSON, adorned with @context, into RDF by the Java command line tool:

```
{
  "@context": "https://github.com/fhircat/jsonld_context_files/…/medicationrequest.context.jsonld",
  "resourceType": "MedicationRequest",
  "id": "medrx002",
  ...
  "dosageInstruction": [
    {
      "doseAndRate": [
        {
          "doseQuantity": {
            "code": "mg",
            "system": "http://unitsofmeasure.org",
            "unit": "mg",
            "value": 500
          },
          "type": {
            "coding": [
              {
                "code": "ordered",
                "display": "Ordered",
                "system": "http://terminology.hl7.org/CodeSystem/dose-rate-type"
              }
            ]
          }
        }
      ],
      "route": {
        "coding": [
          {
            "code": "255559005",
            "display": "Intramuscular (qualifier value)",
            "system": "http://snomed.info/sct"
          }
        ]
      },
      "sequence": 1,
      "text": "Administer 500mg IM as a single dose",
      "timing": {
        "repeat": {
          "count": 1
 } } } ] }
```

Figure 2: FHIR JSON annotated with @context

```
    _:b10 fhir:dosageInstruction [
      fhir:doseAndRate [
        fhir:doseQuantity [
          fhir:Quantity.code     "mg"  ;
          fhir:Quantity.system   <http://unitsofmeasure.org> ;
          fhir:Quantity.unit     "mg" ;
          fhir:Quantity.value    "500"^^xsd:decimal
        ] ;
        fhir:Quantity.type [
          fhir:CodeableConcept.coding [
            fhir:Coding.code     "ordered" ;
            fhir:Coding.display  "Ordered" ;
            fhir:Coding.system   <http://terminology.hl7.org/CodeSystem/dose-rate-type>
          ]
        ] ;
        fhir:route [
          fhir:CodeableConcept.coding [
            fhir:Coding.code     "255559005" ;
            fhir:Coding.display  "Intramuscular (qualifier value)" ;
            fhir:Coding.system   <http://snomed.info/sct>
          ]
        ] ;
        fhir:sequence            1  ;
        fhir:text                "Administer 500mg IM as a single dose";
        fhir:timing [
          fhir:Timing.repeat [
            fhir:Timing.count    1
    ] ] ] ] .
```

Figure 3: RDF Output of Java-based command-line tool

4. Conclusion:

In this pilot effort, we created a JSON-LD-based framework for the FHIR RDF specification implementation, through implementing two JAVA-based JSON-LD tools, i.e., a @context

generator and a command line tool for batch conversion to/from FHIR RDF, based on JSON-LD 1.1. We demonstrated the feasibility of the JSON-LD-based approach to handle shortened predicate names while we are still in the process of implementing the features for handling BNodes and extensions.

The approach of leveraging JSON-LD @context files to translate FHIR Resources and Instance data bi-directionally between FHIR JSON and FHIR RDF has potential to reduce the cost of RDF implementation and maintenance. This also serves to link the FHIR community to other RDF communities. The ability to bidirectionally serialize FHIR artifacts using JSON-LD 1.1 has its own challenges and issues that were addressed with a strategy of pre- or post-processing of instance data and resolving issues with a suite of JSON-LD -based tools. We developed two tools - a JSON-LD @context generator and a command-line tool for batch conversion - with the latest FHIR build and server implementations. The output was validated for a) alignment between the @context files and the ShEx schemas and b) demonstrate to the community the model resulting from issues in the FHIR RDF community. This paves the way for these tools to be integrated into the FHIR build process.

References

1. (HL7). *The HL7 Fast Healthcare Interoperability Resources (FHIR)*. 2021 September 29th, 2021]; Available from: http://www.hl7.org/fhir/.
2. ECMA. *The JSON data interchange syntax*. 2017 September 30th, 2021]; Available from: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/.
3. (W3C). *Extensible Markup Language (XML)*. 2008 September 29th, 2021]; Available from: https://www.w3.org/TR/xml/.
4. (W3C), W.W.W.C. *Resource Description Framework (RDF)*. 2016 September 29th, 2021]; Available from: https://www.w3.org/RDF/.
5. (W3C). *JSON-LD 1.1 - A JSON-based Serialization for Linked Data*. 2021 September 29th, 2021]; Available from: https://w3c.github.io/json-ld-syntax/.
6. Solbrig, H.R., et al., *Exploring JSON-LD as an Executable Definition of FHIR RDF to Enable.* AMIA Annu Symp Proc, 2020. 2020: p. 1140-1149.
7. (W3C). *Shape Expressions Language*. 2019 September 29th, 2021]; Available from: http://shex.io/shex-semantics/.
8. Project, F. *JSON-LD 1.1. Context Generator*. 2021 September 29th, 2021]; Available from: https://github.com/fhircat/org.hl7.fhir.corea.
9. Project, F. *JSON-LD context files at FHIRCat*. 2021 September 29th, 2021]; Available from: https://github.com/fhircat/jsonld_context_files.
10. Project, F. *Shape Expression (ShEx) Generator*. 2021 September 29th, 2021]; Available from: https://github.com/fhircat/org.hl7.fhir.corea.
11. (W3C). *Shape Expressions (ShEx) Schema*. 2019 September 29th, 2021]; Available from: http://shex.io/shex-semantics/#shapes-schema.
12. Project, F. *FHIRCat JSON-LD Command line interface*. 2021 September 29th, 2021]; Available from: https://github.com/fhircat/jsonld-cli.
13. (HL7). *The HL7 FHIR Packages at NPM*. 2021 September 29th, 2021]; Available from: https://www.npmjs.com/package/hl7.fhir.r4.core.