YARRRML + LDES: Simultaneously Lowering Complexity from Knowledge Graph Generation and Publication

Gerald Haesendonck[1,*], Ben De Meester[1], Julián Andrés Rojas Meléndez[1], Dylan Van Assche[1] and Pieter Colpaert[1]

[1] *IDLab, Dep. of Electronics and Information Systems, Ghent University – imec, Zwijnaarde, Belgium*

Abstract

Linked Data Event Streams (LDES) is an advanced Knowledge Graph (KG) publication specification aimed at continuous data source replication and synchronization with benefits such as data entities versioning and history retention while providing a self-descriptive API. However, building an LDES requires a high level of expertise in the Semantic Web ecosystem. In this demo paper, we show how we lower the complexity and need for expertise when using a more advanced KG publication method such as LDES by providing an extension point to YARRRML, a human-friendly way to configure KG generation via RML. Integrated in Matey, an online YARRRML editor, we show how little effort (adding five characters to the YARRRML syntax in the simplest case) allows (re)generating multiple versions of a KG as an Event Stream. As such, this extension provides an easy-to-use starting point for anyone wanting to create an LDES from non-semantic data.

Keywords

Knowledge Graph, YARRRML, LDES

1. Introduction

Linked Data Event Streams (LDES)[1] [1] models publishing changes in Knowledge Graph (KG) data entities structured as an append-only log, allowing clients to replicate the history of a data set and efficiently synchronize with its latest changes. LDES has proven to be useful for integrating and publishing data in different domains such as base registries [1], cultural heritage collections [2], and marine georeferenced locations, boundaries, and regions for scientific and educational purposes [3]. LDES is promoted in a European data interoperability context[2] and the government of Flanders started publishing data as LDES[3].

ISWC 2023 Posters and Demos: 22nd International Semantic Web Conference, November 6–10, 2023, Athens, Greece

*Corresponding author.

gerald.haesendonck@ugent.be (G. Haesendonck); ben.demeester@ugent.be (B. D. Meester); julianandres.rojasmelendez@ugent.be (J. A. R. Meléndez); Dylan.VanAssche@ugent.be (D. V. Assche); pieter.colpaert@ugent.be (P. Colpaert)

0000-0003-1605-3855 (G. Haesendonck); 0000-0003-0248-0987 (B. D. Meester); 0000-0002-6645-1264 (J. A. R. Meléndez); 0000-0002-7195-9935 (D. V. Assche); 0000-0001-6917-2167 (P. Colpaert)

© 2023 Copyright c 2023 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

[1] https://semiceu.github.io/LinkedDataEventStreams/

[2] https://joinup.ec.europa.eu/collection/semic-support-centre/linked-data-event-streams-ldes

[3] https://www.vlaanderen.be/datavindplaats/catalogus?text.LIKE=ldes&types.CONTAINS_ANY=service&order_relevance=asc

However, building an LDES requires a high level of expertise: (i) it requires advanced knowledge of Semantic Web concepts and technologies, and (ii) existing software for publishing LDESes[4] [5] start from version objects already formatted as LDES members, thus requiring an extra step to perform before data can be ingested. This inhibits uptake of the LDES specification and constitutes a problem for advanced KG publication methods in general.

In this demo paper, we show how we lower the complexity and need for expertise when using a more advanced KG publication method such as LDES by providing an intuitive way to configure LDES generation. For this, we rely on the RDF Mapping Language (RML) [4] as a commonly used declarative way to describe a KG construction process. Namely, we extended YARRRML [5], a human-friendly representation for RML. We chose YARRRML as it has multiple extensions already [6, 7] and is used in knowledge graph cloud services such as Google's Enterprise Knowledge Graph[6].

Where others extend YARRRML to stay in sync with the underlying mapping language features, our extension adds no additional features but instead provides an easy-to-use starting point for anyone wanting to create an LDES from existing data.

Integrated in Matey, an online YARRRML editor, we show how little effort (adding five characters to the YARRRML syntax in the simplest case) allows (re)generating multiple versions of KG data entities as an Event Stream.

2. Adding an LDES Extension Point to YARRRML

Our YARRRML extension generates all needed RML rules that comply to an existing method for generating LDES using RML [8]. A full specification and examples are available online at https://rml.io/yarrrml/spec/ldes/. We introduce a single extension point: adding the `ldes` key to the `subjects` collection in YARRRML is enough to define RML mappings generating a valid minimalist LDES with reasonable defaults. Subkeys allow further configuration to customize the resulting LDES.

We include the following properties of an `ldes:EventStream` instance according to the LDES specification[7] `tree:shape`, `ldes:timestampPath`, and `ldes:versionOfPath`. In YARRRML they can be configured using the subkeys `shape`, `timestampPath` and `versionOfPath` respectively. The IRI to identify the `ldes:EventStream` instance can be configured with the `id` sub-key. We do not support LDES fragmentation, pagination and retention policies because they are typically use case specific and can easily be obtained by using existing LDES servers or clients once a basic LDES is available.

Finally, we add the `watchedProperties` sub-key. This key describes which fields in the original data to watch, as only new values in all given fields trigger the generation of a new `tree:member` in the LDES.

The generated RML rules use a function to take the `watchedProperties` values into account when generating a new `tree:member`, as introduced by Van Assche et al [8]. For example,

[4]https://informatievlaanderen.github.io/VSDS-Tech-Docs/docs/LDES_server.html
[5]https://github.com/TREEcg/LDES-Solid-Server
[6]https://cloud.google.com/enterprise-knowledge-graph/docs/overview
[7]https://w3id.org/ldes/specification

SensorID	Timestamp	Temperature
1	2023-01-01T08:00:00	8
2	2023-01-01T08:00:00	9
1	2023-01-01T09:00:00	9
2	2023-01-01T09:00:00	9

Table 1
Example data: sensors and their temperature readings

```
1  subjects:
2    - value: ex:$(SensorID)
3      targets:
4        - [ out.ttl~void, turtle ]
5      ldes:
6        id: ex:myldes
7        watchedProperties: [$(SensorID), $(Timestamp), $(Temperature)]
8        shape: ex:shape.shacl
9        timestampPath: [ex:ts, $(Timestamp), xsd:dateTime]
10       versionOfPath: [ex:hasVersion, ex:$(SensorID)]
```

Listing 1: Example of a `subjects` section with the `ldes` key and properties integrated.

`watchedProperties: [$(Temperature)]` results in a new LDES member only when the value of field `Temperature` in the original data was not recorded before. Listing 1, lines 5 – 10, illustrates how we can generate an LDES from data shown in Table 1.

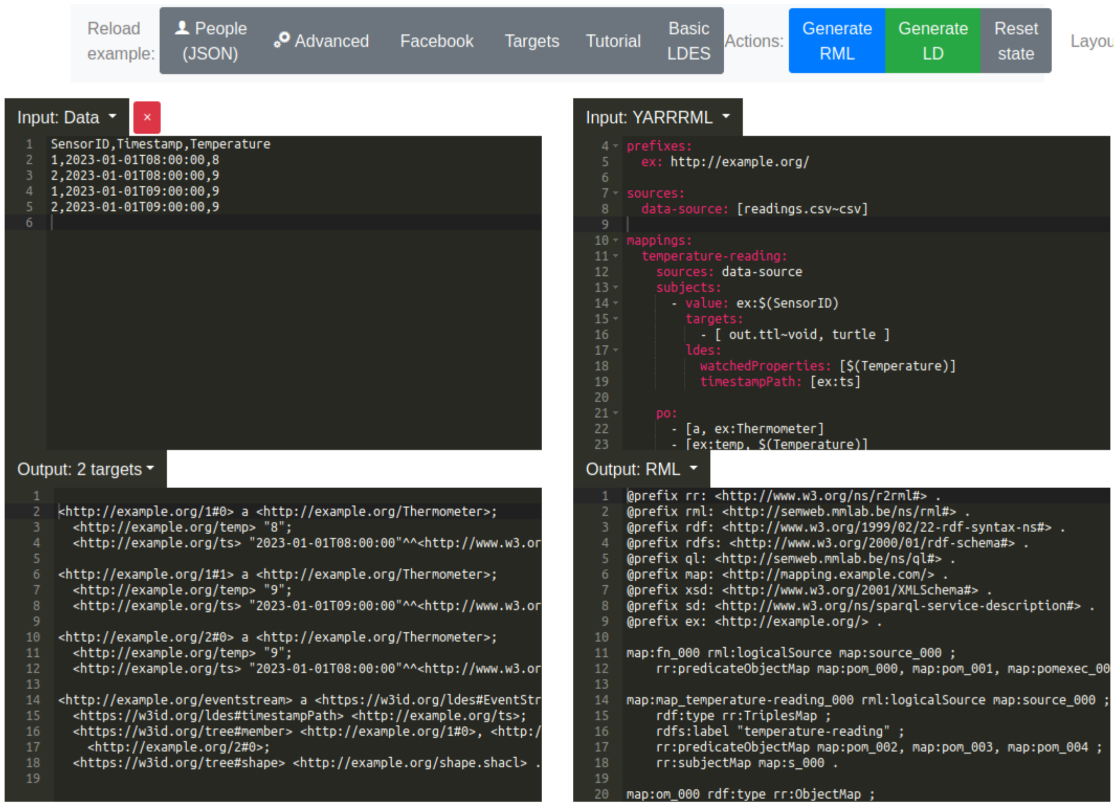
Compared to using RML directly to generate an LDES, our approach reduces technical overhead, e.g., adding an LDES configuration to YARRRML requires maximum 7 additional lines of code, but results in more than 40 additional triples in the RML mapping document[8]. By using `ldes` as key and naming sub-keys close to the corresponding LDES properties, we provide an intuitive and simple way to configure LDES generation. We implemented the LDES extension using the YARRRML Parser[9] which translates YARRRML rules into RML.

3. Matey: Human-friendly KG Generation & Publication

To demonstrate our extension, we extended Matey[10], an open-source browser-based application that helps one write YARRRML rules, publicly available at https://rml.io/yarrrml/matey/.

We added a basic LDES example (loadable by clicking the button `[Basic LDES]`) which loads a CSV file with temperature readings and YARRRML mappings, and generates an LDES based on temperature changes when clicking the `[Generate LD]` button (Figure 1). Because we only want to generate members when the temperature changes, there is no member for sensor 2's

[8]Examples of YARRRML rules and their corresponding RML rules generating LDES can be found at https://github.com/RMLio/yarrrml-ldes-extension/tree/main/ldes, all folders contain links to YARRRML files (.yaml) and their corresponding RML files (.rml.ttl).
[9]https://github.com/RMLio/yarrrml-parser/releases/tag/v1.5.2
[10]https://rml.io/yarrrml/matey/



```
SensorID,Timestamp,Temperature
1,2023-01-01T08:00:00,8
2,2023-01-01T08:00:00,9
1,2023-01-01T09:00:00,9
2,2023-01-01T09:00:00,9
```

```
prefixes:
  ex: http://example.org/

sources:
  data-source: [readings.csv~csv]

mappings:
  temperature-reading:
    sources: data-source
    subjects:
      - value: ex:$(SensorID)
        targets:
          - [ out.ttl~void, turtle ]
        ldes:
          watchedProperties: [$(Temperature)]
          timestampPath: [ex:ts]

    po:
      - [a, ex:Thermometer]
      - [ex:temp, $(Temperature)]
```

```

<http://example.org/1#0> a <http://example.org/Thermometer>;
  <http://example.org/temp> "8";
  <http://example.org/ts> "2023-01-01T08:00:00"^^<http://www.w3.or

<http://example.org/1#1> a <http://example.org/Thermometer>;
  <http://example.org/temp> "9";
  <http://example.org/ts> "2023-01-01T09:00:00"^^<http://www.w3.or

<http://example.org/2#0> a <http://example.org/Thermometer>;
  <http://example.org/temp> "9";
  <http://example.org/ts> "2023-01-01T08:00:00"^^<http://www.w3.or

<http://example.org/eventstream> a <https://w3id.org/ldes#EventStr
  <https://w3id.org/ldes#timestampPath> <http://example.org/ts>;
  <https://w3id.org/tree#member> <http://example.org/1#0>, <http:/
    <http://example.org/2#0>;
  <https://w3id.org/tree#shape> <http://example.org/shape.shacl> .

```

```
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rml: <http://semweb.mmlab.be/ns/rml#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ql: <http://semweb.mmlab.be/ns/ql#> .
@prefix map: <http://mapping.example.com/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix sd: <http://www.w3.org/ns/sparql-service-description#> .
@prefix ex: <http://example.org/> .

map:fn_000 rml:logicalSource map:source_000 ;
    rr:predicateObjectMap map:pom_000, map:pom_001, map:pomexec_00

map:map_temperature-reading_000 rml:logicalSource map:source_000 ;
    rdf:type rr:TriplesMap ;
    rdfs:label "temperature-reading" ;
    rr:predicateObjectMap map:pom_002, map:pom_003, map:pom_004 ;
    rr:subjectMap map:s_000 .

map:om_000 rdf:type rr:ObjectMap ;
```

Figure 1: This Matey demo demonstrates how generating KG versions using the `ldes` key within YARRRML introduces very little additional complexity.

last reading. Clicking `[Generate RML]` shows the corresponding RML mapping rules which are applied on the data. When clicking `[Generate LD]` again, no members are generated at all, since all members are already generated. This demonstrates the feature of adding members to a previously generated LDES without violating the `watchedProperties` constraints. Clicking `[Reset state]` clears the state, after which the LDES can be re-generated.

4. Conclusion

By extending YARRRML to configure how to generate an LDES from existing data we lower the technical complexity of a task that requires domain-specific knowledge. Future research is needed to go beyond basic LDES generation and explore more advanced features such as fragmentation. We expect this type of work to (i) increase LDES uptake and (ii) inspire other (YARRRML-like) extensions to lower complexity for any KG generation or publication method.

Acknowledgments The described research activities were supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10), the Flemish Smart Data Space

(Flemish Government, Digital Flanders and RRF project VV073), and the imec ICON project BoB (Agentschap Innoveren en Ondernemen project nr. HBC.2021.0658).

References

[1] D. Van Lancker, P. Colpaert, H. Delva, B. Van de Vyvere, J. Rojas Meléndez, R. Dedecker, P. Michiels, R. Buyle, A. De Craene, R. Verborgh, Publishing Base Registries as Linked Data Event Streams, in: Web Engineering (ICWE), 2021. doi:10.1007/978-3-030-74296-6_3.

[2] B. Van de Vyvere, O. V. D'Huynslager, A. Atauil, M. Segers, L. Van Campe, N. Vandekeybus, S. Teugels, A. Saenko, P.-J. Pauwels, P. Colpaert, Publishing cultural heritage collections of ghent with linked data event streams, in: Metadata and Semantic Research, 2022.

[3] B. Lonneville, H. Delva, M. Portier, L. V. Maldeghem, L. Schepers, D. Bakeev, B. Vanhoorne, L. Tyberghein, P. Colpaert, Publishing the marine regions gazetteer as a linked data event stream, in: Proceedings of the Joint Ontology Workshops 2021 Episode VII: The Bolzano Summer of Knowledge co-located with the 12th International Conference on Formal Ontology in Information Systems (FOIS 2021), and the 12th International Conference on Biomedical Ontologies (ICBO 2021), Bolzano, Italy, September 11-18, 2021, 2021.

[4] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, in: Proc. of the 7th Workshop on Linked Data on the Web, 2014.

[5] P. Heyvaert, B. De Meester, A. Dimou, R. Verborgh, Declarative Rules for Linked Data Generation at your Fingertips!, in: The Semantic Web: ESWC 2018 Satellite Events, 2018.

[6] D. Van Assche, T. Delva, P. Heyvaert, B. De Meester, A. Dimou, Towards a more human-friendly knowledge graph generation & publication, in: International Semantic Web Conference (ISWC) 2021: Posters, Demos, and Industry Tracks, 2021.

[7] B. Steenwinckel, G. Vandewiele, I. Rausch, P. Heyvaert, R. Taelman, P. Colpaert, P. Simoens, A. Dimou, F. De Turck, F. Ongenae, Facilitating the analysis of covid-19 literature through a knowledge graph, in: J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web – ISWC 2020, 2020.

[8] D. Van Assche, S. M. Oo, J. A. Rojas, P. Colpaert, Continuous generation of versioned collections' members with RML and LDES, in: Proc. of the 3rd International Workshop on Knowledge Graph Construction (KGCW 2022), 2022.