

# LDESTS: Enabling efficient storage and querying of large volumes of time series data on Solid pods

Tom Windels<sup>1,\*</sup>, Wout Slabbinck<sup>1</sup>, Pieter Bonte<sup>1,2</sup>, Stijn Verstichel<sup>1</sup>, Pieter Colpaert<sup>1</sup>, Sofie Van Hoecke<sup>1</sup> and Femke Ongenae<sup>1</sup>

<sup>1</sup>IDLab, Ghent University – imec, Belgium

<sup>2</sup>Department of Computer Science, KU Leuven Campus Kulak, Belgium

## Abstract

The Solid ecosystem provides a good foundation for the decentralised Web. However, the current document-based implementations of the Solid specification lack support to efficiently interact with high volumes of time series data. Therefore, in this demo, we present a technique, called LDESTS, to more efficiently store and query time series data across Solid pods. LDESTS provides a data model for individual streams, and enables enforcing constraints on top of individual stream fragments, to achieve efficient data fragmentation and querying.

## Keywords

Time series data, Decentralized Web, Solid, Efficient data storage & querying

## 1. Introduction

The current digital ecosystem mainly consists of centralised solutions, in which users trust various services with their data. Whilst this comes with benefits related to application design, there are significant, non-technical, drawbacks too [1]. The main drawbacks are loss of control over the data by the users, leading to various privacy and legislative concerns, e.g. GDPR, and replication of user data across multiple centralized silos, leading to inconsistencies and unnecessary economic investments in data harvesting and storage. As such, a rapid paradigm shift can be noted towards decentralized storage of user data, which is beneficial for both users and owners of these digital services to resolve the aforementioned drawbacks. To realize this shift, the Solid specification and ecosystem<sup>1</sup> was proposed in which every user controls their own private storage in the form of one or more Solid Pods, guarding all public and private data they or others create about them. These data vaults allow individuals to decide at every moment to which people and organizations they selectively grant read or write access and to which specific data.

Solid mostly uses RDF documents to represent the data it stores, as this allows users and developers to represent all sorts of data in relevant domain-specific models in an interoperable

---


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

\*Corresponding author.

✉ tom.windels@ugent.be (T. Windels); wout.slabbinck@ugent.be (W. Slabbinck); pieter.bonte@kuleuven.be (P. Bonte); stijn.verstichel@ugent.be (S. Verstichel); pieter.colpaert@ugent.be (P. Colpaert); sofie.vanhoecke@ugent.be (S. V. Hoecke); femke.ongenae@ugent.be (F. Ongenae)



© 2023 Copyright 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)

<sup>1</sup><https://solidproject.org/TR/>

fashion. Current implementation of the Solid specification mainly use the Linked Data Platform (LDP) ontology<sup>2</sup> to mimic a UNIX-like data structure, where documents are structured in various folders. It further imposes no extra restrictions on data structure, leaving room for flexibility [2].

The current document-centric interpretation of Solid, wherein a pod is a hierarchy of Linked Data documents, is not the most efficient solution for all types of data and services. With the boom of the Internet of Things (IoT), time series data has become increasingly prevalent. Examples include data gathered by wearable devices or sensors found in a smart home. These data streams can accumulate to large volumes of rapidly changing data over time, resulting in the need to store this data efficiently. As this stream of time series data is stored in a remote location, and most applications want to query over this data, the network traffic should be kept to a minimal for these applications to work efficiently. Therefore, this demo presents a technique, called LDESTS (Linked Data Event Streams for Time Series), to more efficiently store and query time series across Solid pods.

## 2. Preliminaries

Implementations of the Solid specification targeting data streams have recently been proposed, i.e. Linked Data Event Streams (LDES) in LDP [3]. This technique is based on the LDES specification<sup>3</sup>, which is an ontology for representing collections of immutable objects, like sensor measurements. It fragments the stream across multiple documents, which in turn are fragmented across multiple folders, based on a given metric, e.g. timestamps, and optionally different object versions. By splitting up the stream in smaller documents, LDES allows clients to limit the amount of data gathered. Research [4, 5] has been performed to decide the optimal number of samples stored per document based on the velocity of the stream and application requirements, i.e. performed queries. To allow the resulting fragments to be queryable by time, TREE hypermedia is used<sup>4</sup>. This ontology provides relations between the various fragments and the stream, allowing query processors to selectively query the requested fragments.

Even with these optimizations, LDES in LDP causes a lot of overhead and network load to store and query rapidly changing time series data, especially for data streams where the difference between similar data points is small, e.g. a sensor from a user's wearable, measuring the same type of measurement with similar properties, but at a different point in time. When an application is only interested in a specific sample, it still needs to fetch a whole (possibly large) document. Moreover, if the application requires a large range of samples, a lot of fetches across various documents are needed. Due to the potentially flexible nature of individual data points, querying over data can only optimally choose its fragments based on the requested time through TREE relationships, and not the properties of these data samples. The verbosity of RDF additionally adds a lot over overhead and network load, as the samples are surrounded by repeated metadata describing their semantics. LDESTS aims to solve these shortcomings.

---

<sup>2</sup><https://www.w3.org/TR/ldp/>

<sup>3</sup><https://semiceu.github.io/LinkedDataEventStreams/>

<sup>4</sup><https://treecg.github.io/specification/>

### 3. LDESTS overview

Time series data typically observes a single metric, leaving all other attributes in the RDF representation unchanged. For example, for two consecutive measurements of the same heart rate sensor, only the heart rate value and timestamp changes, while all the other metadata about the observation, e.g. associated sensor or location, remains the same. Other scenarios include time series data streams that are generated by multiple sources, each measuring the same metric, and independently generating a similar repeating RDF structure for the sample's properties, but associate a different value to the sample's source. An example would be the user's smartphone and smartwatch obtaining accelerometer values. To more efficiently represent the data, LDESTS avoids these superfluous repetitions of common properties by defining an optimized data model per stream. The LDESTS data model uses the Shapes Constraint Language (SHACL)<sup>5</sup> ontology to define the various properties of a data sample combined with its specific role. There are three possible roles a sample property can have:

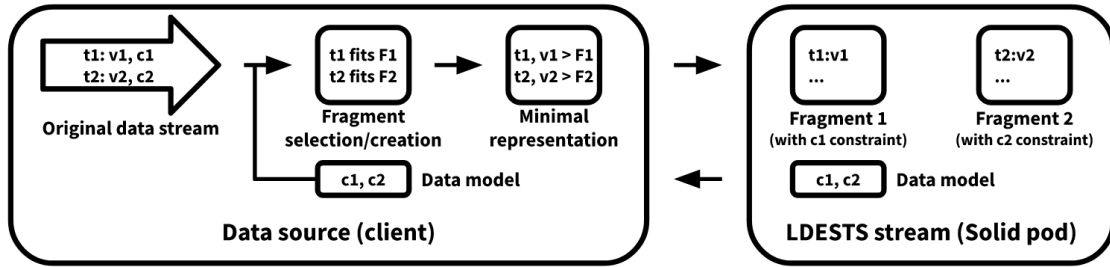
- **Identifier property:** Denotes the sample's predicate containing the timestamp. For example, in a sensor measurement using the Smart Applications Reference ontology (SAREF)<sup>6</sup>, this would be `saref:hasTimestamp`. Exactly one such property is allowed per data model.
- **Constant property:** Declares a property with which either a single constant value is associated for every sample in the stream (the source property of the data, for example), or a set of constant values that vary on a per-sample basis (the axis in a multiple-axes measurement scenario, for example). The latter case can be used for further fragmentation of the stream, improving the performance of property specific queries.
- **Variable property:** Declares a value that differs per sample. This is often the metric being observed in this time series data stream.

Solid imposes little restriction on how the contents of a user's pod should be hierarchically structured. Therefore, we propose a custom hierarchy that allows LDESTS to optimally exploit the properties of the original data for efficiency gains. The components making up this hierarchy consists out of the individual fragments as LDP resources adhering to stream-specific constraints, and the LDP container housing these fragments. The LDP container's metadata declares the stream's properties, e.g. the data model associated with the data described above, and the various fragments making up the stream. Fragments have their own set of restrictions, further refining the stream's data model to the exact type found in that fragment, and provide TREE relationship information as well. A diagram depicting this flow can be seen in Figure 1. As the exact data model per fragment is known prior to querying the actual data, a specialised query processor can choose to ignore specific fragments based on the query directly. The fragments house the set of data samples themselves by tightly formatting the sample's properties in a string literal. Constants differing per sample are represented with an index that can be used with the set of values defined in the data model for that property, while variables are inserted directly.

---

<sup>5</sup><https://www.w3.org/TR/shacl/>

<sup>6</sup><https://saref.etsi.org/core/v3.1.1/>



**Figure 1:** LDESTS converting the original data stream and fragmenting it on the Solid Pod

## 4. Demonstrator scenario and results

LDESTS is demonstrated based on a homecare use case, in which a patient is being monitored during their daily routines using various sensors, e.g. a wearable measuring accelerometer and heart rate, that generate continuous and vast streams of data. LDESTS is used to efficiently store the data in the patient’s Solid pod. This pod can then be queried by parties authorized by the patient, e.g. a care organization, to gain insight into the condition of the patient. For the demonstration, the Data Analytics for Health and Connected Care (DAHCC) Open Dataset is used [6], which provides real-world collected sensor data from 30 persons in a home setting, already annotated as RDF using the SAREF ontology<sup>7</sup> to indicate the source, type of metric and timestamp, amongst other things for each measurement. The demo replays the data of one of the selected patients in the DAHCC dataset, at the same rate as it was collected, and stores it in a Solid pod using LDESTS. The data model in the created stream is provided with multiple possible sources and the various axes of the sensor values as constant properties, the timestamp as the identifier property and the sensor value itself as the only variable property. The stream is fragmented according to the data source. With an input of 14.8 MB worth of triples, representing measurements done over the span of half an hour, the resulting stream only uses 378 kB distributed across the stream definition and 6 fragments, resulting in a 97% reduction. The demo code (with video) is available at <https://github.com/SolidLabResearch/LDESTS-demo>.

## Acknowledgments

This research was partly funded the Flemish Government under the “Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” programme, the SolidLab Vlaanderen project (Flemish Government, EWI and RRF project VV023/10) and the FWO Project FRACTION (Nr. G086822N).

## References

- [1] R. Verborgh, Re-decentralizing the Web, for good this time, in: *Linking the World’s Information: A Collection of Essays on the Work of Sir Tim Berners-Lee*, 2022.

<sup>7</sup><https://dahcc.idlab.ugent.be/>

- [2] R. Dedecker, et. al., What's in a Pod? – a knowledge graph interpretation for the Solid ecosystem, in: Proc. of the 6th QuWeDa Workshop at ISWC, 2022.
- [3] W. Slabbinck, et. al., Linked data event streams in Solid LDP containers, in: Proc. of the 8th MEPDaW Workshop at ISWC, 2022.
- [4] B. Van de Vyvere, et. al., Publishing cultural heritage collections of ghent with linked data event streams, in: Proc. of MTSR, 2021.
- [5] H. Delva, et. al., Geospatially partitioning public transit networks for open data publishing, *Journal of Web Engineering* (2021).
- [6] B. Steenwinckel, et. al., Data Analytics For Health and Connected Care: Ontology, Knowledge Graph and Applications, in: Proc. of PervasiveHealth, 2022.