VoCaLS: Vocabulary & Catalog of Linked Streams

Riccardo Tommasini[1,a], Yehia Abo Sedira[1,b], Daniele Dell'Aglio[2], Marco Balduini[1,a], Muhammad Intizar Ali[3], Danh Le Phuoc[4], Emanuele Della Valle[1,a], Jean-Paul Calbimonte[5]

[1]Politecnico di Milano, DEIB, Milan, Italy
[a]{name.lastname}@polimi.it | [b]yehiamohamed.abosedera@mail.polimi.it
[2]University of Zurich, Zurich, Switzerland
dellaglio@ifi.uzh.ch
[3]Insight Center for Data Analytics, National University of Ireland, Galway, Ireland
ali.intizar@insight-centre.org
[4]Technichal University of Berlin, Berlin, Germany
danh.lephuoc@tu-berlin.de
[5]University of Applied Sciences and Arts Western Switzerland, Sierre, Switzerland
jean-paul.calbimonte@hevs.ch

Abstract. The nature of Web data is changing. The popularity of news feeds and social media, the rise of the Web of Things, and the adoption of sensor technologies are examples of streaming data that reached the Web scale. The different nature of streaming data calls for specific solutions to problems like data integration and analytics. There is a need for streaming-specific Web resources: new vocabularies to describe, find and select streaming data sources, and systems that can cooperate dynamically to solve stream processing tasks. To foster interoperability between these streaming services on the Web, we propose the Vocabulary & Catalog of Linked Streams (VoCaLS). VoCaLS is a three-module ontology to (i) publish streaming data following Linked Data principles, (ii) describe streaming services and (iii) track the provenance of stream processing.

1 Introduction

Streams have become increasingly more relevant in several scenarios, including sensor data analytics, social networks, or the Internet of Things. Handling the variety and velocity dimensions together has proven to be hard, and the Semantic Web community has answered to these challenges producing languages, models, systems, and benchmarks under the *Stream Reasoning* umbrella [12]. Despite the progress that Stream Reasoning efforts constitute, the interest in exploring stream publication and consumption mechanisms on the Web has only recently gained attention [14,23]. As opposed to traditional static and stored RDF data, streams are produced and consumed in a different way, focusing on the liveness and dynamics of the data, and often requiring alternative protocols and mechanisms for dealing with data velocity. Systems that consume streams for

processing (e.g., reasoning, filtering, learning, event detection) require standards for interchanging data about the streams, including endpoint information, processing capabilities, data structure, pull and push retrieval options, and querying specificities.

Different studies have partially tackled these problems in the past [4,24], although there is still no general agreement on a shared set of principles, as it is the case with *static* Linked Data. A set of challenges and requirements regarding the availability of streams on the Web has been presented in [14], providing a road-map towards a Web of Data Streams. Further examples in this scope include generic RESTful service interfaces for streaming data, such as in [6,24]; or the RSP Service Interface [3], providing a programming API for continuous query engines. Other approaches adopted a Linked Data-based publishing strategy [20], although in practice they show the inadequacy of static data publishing for this purpose. Also, systems like TripleWave [22] allow live provisioning of RDF stream data through push and pull mechanisms, thus generating live RDF stream endpoints. Nevertheless, in all these cases, metadata about the streams and their access points and methods have used ad-hoc description vocabularies or project specific ontologies.

This paper presents a (i) set of requirements that take into account recent challenges and issues; (ii) it describes a **Vo**cabulary for **Ca**taloging and **L**inking **S**treams and streaming services on the web (VoCaLS[1]). Last but not least, the paper (iii) draws a road map towards the creation of a catalog that would make streams discoverable, accessible, and reusable. VoCaLS is an emerging resource that standardizes the mechanisms to publish and consume semantic streams on the Web. This includes not only the publication of streams but also the consumption and processing, regardless of implementations details and design choices of different RDF Stream Processing (RSP) and Stream Reasoning systems and languages. This vocabulary constitutes a foundational step towards the long-term goal of allowing Web-centered interactions among RDF Stream processing services. VoCaLS has been engineered as a collaborative effort, following the discussions and results of the work of the W3C RSP Community Group[2]. The vocabulary has been made openly available through a permanent URI, it has been submitted to the Linked Open Vocabularies (LOV) repository [26], it is published under a CC-BY 4.0 license, and its documentation is made available through the Widoco toolset [17]. Furthermore, the ontology itself has been designed in a generic manner, so that it can be reused and combined with domain-specific and technology-specific vocabularies.

The remainder of the paper is structured as follows: we in Section 2, we discuss some motivating use-cases. We present the requirements analysis in Section 3. We show reused vocabularies Section 4 before introducing the VoCaLS modules and how to combine them with other vocabularies in Section 5. Section 6 describes the related work, while Section 7 concludes the paper and presents the roadmap.

[1] VoCaLS URI: `https://w3id.org/rsp/vocals#`
[2] `https://www.w3.org/community/rsp/`

2 Use-Cases

In this section, we describe three use-cases that motivate the design and the adoption of a vocabulary for describing streams and streaming services.

2.1 Stream Discovery & Selection in Smart Cities

Smart cities are one of the early adopters of IoT technologies and subsequently of stream processing. Since many publisher and consumers coexist in city sensor networks, middlewares and semantic technologies are commonly used to enable automated discovery and integration [16,18]. The publishers purpose is making the stream findable to the middleware, while the consumers intent is finding and selecting the proper streams that solve a given information need. Therefore, urban data streams are enriched with semantic annotations; their selection is automated using technologies that interpret descriptions.

The most significant hindrance to this approach is centralization. Middlewares often rely on a central repository of stream descriptions, because interoperability requires a standardized interfaces. With the adoption of a shared vocabulary, the decentralized automated discovery would be possible at the Web scale. As for Linked Datasets, stream provisioning services will be able to exchange descriptions to agents requests on-demand, reducing the middleware load.

2.2 Streaming Service Discovery & Federation

Federating a query to a remote stream endpoint is a desirable feature for a situation where data are naturally distributed. This is particularly true for streaming sources, where the time required to gather the data might overcome the responsiveness requirements posed by an information-need.

To make RDF Stream Processing (RSP) federation work in practice, we should follow the example of the Linked Data community. SPARQL query federation relies on protocols and dataset descriptions such as VoID [1] and DCat [21]. To support interactions between query engines on the Web, we need (i) standardizing the language, (ii) fixing the protocols and, last but not least, (iii) agreeing on the vocabulary to describe various resources.

Looking at the RSP state-of-the-art, (i) Dell'Aglio et al [13] provided a reference model that unifies RSP languages and reconcile the execution semantics of existing RSP engines [5,19]. (ii) Balduini et al [3] designed a set of RESTful APIs that regulate how to interact with an RDF Stream Processing (RSP) engine in a declarative way.

The missing building block is (iii) the adoption of a shared vocabulary to publish the streams and describe the services. Languages, APIs, and vocabularies together would foster interoperability between different implementation and instances of RSP Services, which up to this point has not happened yet.

2.3 Reproducibility of Empirical Research

Benchmarking is a relevant research topic within the Stream Processing community. Ontologies, datasets, and queries were proposed so far as benchmarks to evaluate engines performance and processing capabilities. Recent efforts tried to formalize an experimental environment that could make empirical research systematic and foster experiment reproducibility [25]. In this context, cataloging available streams, profiling the features of the engine, and tracking the provenance of the experiment as used queries and obtained results, would improve the research outcomes, fostering reproducibility and repeatability.

3 Requirements Analysis

In this section, we present our requirement analysis for VoCaLS. Building on our previous results [13,14,23], we have identified a series of challenges that have to be addressed to comply with the needs of stream processing scenarios. This analysis also takes into account the use cases discussed in the W3C RSP Community Group, as well as the general requirements[3]. We organized the challenges in three main topics: Publication & discovery, Access & processing, and Provenance & licensing.

Publication & discovery. This aspect refers to the description of streams and streaming services, shared according to the Linked Data principles [7], for the creation of catalogs and discovery endpoints. In particular, a *stream description* should (C1) characterize the contents of a (RDF) stream content and (C2) describe the characteristics of the stream source. Moreover, a *streaming service description* should (C3) describe available endpoints from which streams can be accessed/processed/generated. In this scope, such a vocabulary should be able to answer to questions such as:

- What is the identifier/address of a stream?
- Who created, maintains and/or publishes the stream?
- How frequently is the stream content produced?
- Where is located the vocabulary describing the stream content?

Access & processing. These challenges focus on descriptions of protocols and APIs to obtain data from the streams, communicate with the streaming services and manipulate the data. To serve the necessity of managing streaming data in real-time, it is crucial to (C4) describe the capabilities of streaming services, such as stream processing engines and reasoners, in terms of their features (e.g. available operators, entailment regimes, etc.). It is also needed to (C5) maintain the order of elements in the stream. Moreover, it is important to (C6) allow the selection of stream partitions and windows, which can be dumped, transmitted or filtered, enabling time-series analysis and replay. Questions relevant to this scope include:

[3] `https://w3id.org/rsp/requirements`

- Where can I access the live stream?
- Where can I access the stream as a static dataset?
- Where can I access the history of the historical stream as a static dataset?
- Where can I access the stream starting from a point in the past?
- What is the preceding element of a given stream element?

Provenance & Licensing. These challenges refer to tracking the transformations that involve streaming data, and those that occur on the streams, as well as contracts that regulate data access by actors involved in such transformations (C7). Questions in this scope include:

- How can we describe the process that generated the stream?
- How can we describe the process that generated the stream windows?
- Which datasets or streams were used to derive the stream?

To support our requirement analysis, we investigated how the community perceives the challenges, we identified. Therefore, we designed a survey[4] that aims at gathering more precise information about the perception of stream metadata needs, and the relation between Streaming Data and Linked Data. The hypothesis we started from was that current vocabularies for static/stored (Linked) Data are not enough to satisfy scenarios involving streaming data. We therefore formulated 18 questions (not counting those with multiple options), which aim at (i) investigating the potential impact of a vocabulary resource (3 questions), (ii) probing the relevance of specific challenges (8 questions), and (iii) quantifying the knowledge of the survey respondents on the indicated themes (7 questions).

We collected 34 answers[5] mainly from the Stream Reasoning and Linked Data communities. We asked the participants to self-evaluate their competences on Linked Data, Stream Processing, and Stream Reasoning. Moreover, we cross-checked their answers with simple technical questions. As a result, the survey respondents showed an equally distributed knowledge of the two domains (Linked Data and Stream Reasoning). Although only 11% of them declared to be confident with vocabularies like DCAT, VoID, and DCTerms, they all acknowledged the Linked Data principles.

From the investigation, it emerges that for 55% of the respondents Streaming Data are highly relevant research-wise (5 points on a maximum of 5). Moreover, 35% of the respondents have a high interest in Linked Data. This suggests that being the respondents equally distributed between the two communities, Streaming Data is relevant to the Linked Data community too. The survey also shows that 51% respondents evaluated the challenges we presented at least as *important* or *crucial*. Nevertheless, most of the challenges resulted to be unsatisfiable or not entirely addressed by the most popular Linked Data vocabularies, e.g., VoID. Indeed, excepting C3 and C7, for which respectively 45% and 51% of the respondents positively answered, all the remaining challenges collected mostly uncertain answers (i.e., Maybe/I Don't Know). All the results are reported in Table 1.

[4] `https://ysedira.github.io/vocals/survey.md`
[5] `https://goo.gl/zsEJXe`

	Relevance					VoID Adequacy		
Challenges	U	M	IDK	I	C	Yes	No	IDK/Maybe
C1	0.00%	6.06%	3.03%	30.30%	60.61%	18.18%	15.15%	66.67%
C2	0.00%	9.09%	21.21%	54.55%	15.15%	27.27%	18.18%	54.55%
C3	0.00%	6.06%	6.06%	54.55%	33.33%	45.45%	9.09%	45.45%
C4	0.00%	18.18%	6.06%	33.33%	42.42%	3.03%	45.45%	51.52%
C5	0.00%	9.09%	12.12%	63.64%	15.15%	18.18%	18.18%	63.64%
C6	6.06%	9.09%	27.27%	48.48%	9.09%	33.33%	9.09%	57.58%
C7	0.00%	21.21%	27.27%	45.45%	6.06%	51.52%	9.09%	39.39%

Table 1. Challenges Relevance and VoID adequacy to solve them. Legend: U:Useless; M:Marginal; IDK:I Don't Know; I:Important; C:Crucial.

Last but not least, we considered that one of the main differences between streaming and static data is related to the protocols required to access them [23]. Our idea was confirmed by 61% of the survey respondents that agreed (29%) or strongly agreed (32%) with our statement.

Furthermore, we specifically asked our respondents to evaluate with a score from 1 to 5 the nature of streaming data as pull-based (1) or push-based (5). 21 % of the respondents consider streaming data as naturally push-based against the 9% that consider them as pull-based. 35% percent of the respondents are inclined to push-based (4) against the 6% (2). The remaining 29% of the respondents expressed a neutral vote (3).

Supported by the results we acquired, we formulate the following requirements for our vocabulary to satisfy. VoCaLS must:

R1 enable the description of streams, i.e., characterizing their content, relevant statistics, and the license of use;
R2 enable the description of streaming services, i.e., characterizing their capabilities, their APIs, and the license of use;
R3 enable historical stream processing/analysis and replay, i.e., allowing stream storage and dumping of stream samples;
R4 enable provenance tracking at any level, i.e., characterizing stream (a) creation, (b) publication, and (c) storage; but also denoting manipulation and management concerning to existing theoretical frameworks;
R5 tame velocity for streaming data management, i.e., prioritize push-based content provisioning to pull-based one, and encouraging the adoption of an active stream processing paradigm;
R6 tame variety for streaming data management, i.e., do not bind the specification to any domain specific vocabulary, e.g., SSN [11] for IoT or SIOC [8] for Social Media, and to any specific data models, e.g., RDF Streams.

In general, the survey results show that the requirements we collected justify the introduction of a new vocabulary dedicated to describing the different aspects that are only partially covered by existing vocabularies. The limitations of current vocabularies are related to the fact that streaming data requires different (potentially multiple) access methods, going beyond pull and one-off query mechanisms. It is also evident that unlike traditional Linked Data, RDF

streams cannot be just de-referenced. Indeed, data items have to be recovered in a streaming fashion or at least partitioned in windows. Finally, RDF stream services often have different features and operators that may result in different types of streaming results. Differences among systems in this respect have been studied in the past [13], although there are no vocabularies available that represent this type of information.

4 Background & Vocabulary Reuse

In this section, we describe related vocabularies that VoCaLS reuses, and those that inspired part of its design.

Dataset description vocabularies were designed primarily with static and stored (linked) data in mind. However, they provide metadata descriptions for any sort of datasets published on the Web. Indeed, they have found a wide use not only within the Semantic Web community but also the wider Open Data movements all over the world.
The **Data Catalog Vocabulary (DCAT)** [21] is an RDF vocabulary designed to foster interoperability among data web published catalogs. It focuses on describing how datasets are accessible and distributed. From DCAT, we extended the notions of *Distribution*, *Dataset* and *Catalog*.
The **Dublin Core Terms (DCterms)**[6] is the first vocabulary made to describe both physical and Web resources, and provides fifteen generic terms to ad dataset metadata. The properties *title, creator, subject, description* can be used in combination with VoCaLS, since they do not directly refer to datasets.
The **Vocabulary of Interlinked Datasets (VoID)** [1] aims at describing RDF datasets and cross-dataset links. VoID's use-cases comprise dataset discovery, selection and query optimization. From VoID, properties related to *data-dumps*, *features*, and specific static resources can be used.
The **SPARQL Service Description (SD)**[7] is a W3C recommendation that contains the necessary terminology to describe SPARQL endpoints and, thus, it is relevant w.r.t. VoCaLS Service Description. VoCaLS does not extend SPARQL-related terms from SD on purpose, since we consider more appropriate to maintain the specification unbiased. However, SD can be reused in several streaming scenarios, due to the similarity between some SPARQL and RSP operations. Indeed, SD properties like entailment regime, and supported languages can be used with VoCaLS.
The **Provenance Ontology (PROV-O)** allows to describe agent-entity-activity relations that captures the semantics of transformations. VoCaLS Provenance modules extends PROV-O adding some minimal nomenclature that simplifies the usage of this pattern in the context of streaming data processing.

[6] `http://dublincore.org/documents/dcmi-terms/`
[7] `https://www.w3.org/TR/sparql11-service-description/#`

We discuss now two vocabulary drafts that were then merged into VoCaLS. In fact, we decided to re-design the whole vocabulary, considering the limitations of these two early attempts:
The **Vocabulary of Interlinked Streams (VoIS)** [23] extends VoID around four challenges—Discovery, Access, Recall, and Provenance—to model stream interlinking. VoIS is an ontology that provides the classes to publish streams attaching a static description, and allows defining several access methods (e.g., WebSockets, RSP engine) that can be attached to the description.
The **Web Stream Processing (WeSP)** [14] refines the idea of SLD, and is currently implemented in a set of systems, such as TripleWave [22] and CQELS. WeSP comprises a vocabulary to exchange RDF streams on the Web that is built on top of DCAT and the SPARQL SD. It allows describing the stream; some models for stream serializations, and communication protocols.

Given that both VoIS and WeSP have still little or no adoption, and considering their compatibility issues with other vocabularies such as DCAT, we preferred not to reuse terms from them and design VoCaLS from scratch.

5 Vocabulary of Linked Streams

In this section we present the design of VoCaLS. The vocabulary is organized in three modules: VoCaLS Core, which describes the core elements of the vocabulary, VoCaLS Service Description, which describes RDF stream service descriptions, and VoCaLS Provenance, focused on streaming data transformation and manipulation. We will introduce each module separately, along with illustrative examples.

5.1 Core Vocabulary

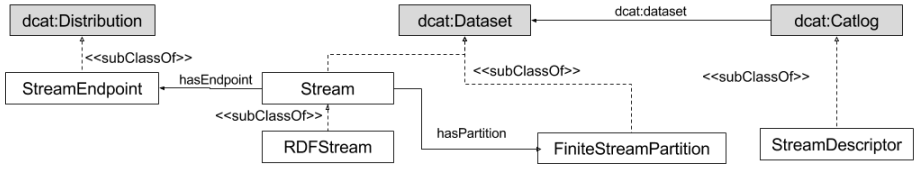

Fig. 1. VoCaLS Core module

VoCaLS Core concepts are based on an extension of DCAT to represent streams on the Web. As depicted in Figure 1 and presented in Listing 1.1, the model introduces the basic abstractions to represent streams. A (i) `vocals:StreamDescriptor` is a document accessible via HTTP that holds metadata about the stream and its contents. A (ii) `vocals:Stream` represents a Web stream, i.e., an unbounded sequence of time-varying data elements [13] that might be findable and accessible on the Web, and which can be consumed via a (iii) `vocals:StreamEndpoint`. An example of Stream Endpoint is available in

Listing 1.1, line 8. Finally, a (iv) `vocals:FiniteStreamPartition` is a portion of the stream available for regular Linked Data services to access and process its content.

```
<> a vocals:StreamDescriptor , vsd:CatalogService ;
   dcat:dataset :MilanTrafficStream .
:MilanTrafficStream a vocals:RDFStream ;
  vocals:hasEndpoint :MilanTrafficStreamEndpoint ;
  dct:title "Milan Traffic Stream"^^xsd:string ;
  dct:publisher <www.3cixty.eu>;
  dct:description "Aggregated stream produced by traffic sensors in Milan
      "^^xsd:string .
:MilanTrafficStreamEndpoint a vocals:StreamEndpoint ;
  dct:license <https://creativecommons.org/licenses/by-nc/4.0/> ;
  dct:format frmt:JSON-LD ;
  dcat:accessURL "ws://example.org/traffic/milan".
```

Listing 1.1. An RDF Stream and Endpoint descriptions using VoCaLS. Prefixes have been omitted.

VoCaLS Core module enables stream producers to publish metadata to describe streams (R1). It also provides a way to represent and describe finite stream partitions that facilitate historical stream processing (R3). With such metadata available on the Web, consumers can discover and select streams relevant to their tasks: for instance, a consumer can retrieve all available endpoints for a given stream, as per Listing 1.2.

```
SELECT ?endpoint
WHERE {
 :traffic_stream a vocals:Stream ;
    vocals:hasEndpoint ?endpoint .
}
```

Listing 1.2. SPARQL query retrieving a StreamEndpoint.

5.2 Streaming Service Description

VoCaLS Service Description focuses on offering a way to publish metadata related to various streaming services and their capabilities, enabling consumers to discover and select services suitable to their needs. The `vsd:StreamingService` is an abstraction to represent a service that deals with data streams of any type. Continuous query engines, stream reasoners, and RDF stream publishers are valid examples.

As depicted in Figure 2, three classes of RDF streaming services were identified, although others could be added if needed: (i) `vsd:CatalogService`, a service that may provide metadata about streams, their content, query endpoints and more. (ii) `vsd:PublishingService`, which represents a service that publishes RDF streams (e.g. TripleWave in Listing 1.3), possibly following a

Linked Data compliant scheme, and (iii) `vsd:ProcessingService`, which models a stream processing service that performs any kind of transformation on streaming data, e.g. querying, reasoning, filtering, as in Listing 1.5. These services include the possibility of specifying push-based publishing paradigms (R5).

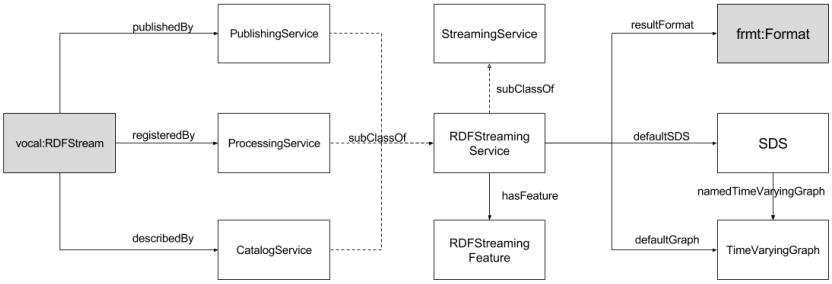

Fig. 2. VoCaLS Service Description classes subset describing RDF streaming services.

```
:trplwv1 a vsd:PublishingService ;
  vsd:hasFeature vsd:replaying ;
  vsd:hasFeature vsd:filtering ;
  vsd:resultFormat frmt:JSON-LD .
```

Listing 1.3. RSP Publishing description using VoCaLS. Prefixes omitted.

Figure 3 shows how VoCaLS Service Description can be used to describe different services, associating each service to the various `vsd:RDFStreamingFeature` it provides, such as what is the reporting policy [13] used by the query engine, which type of time is control is applied, and what the timestamp associated with each stream element represents.

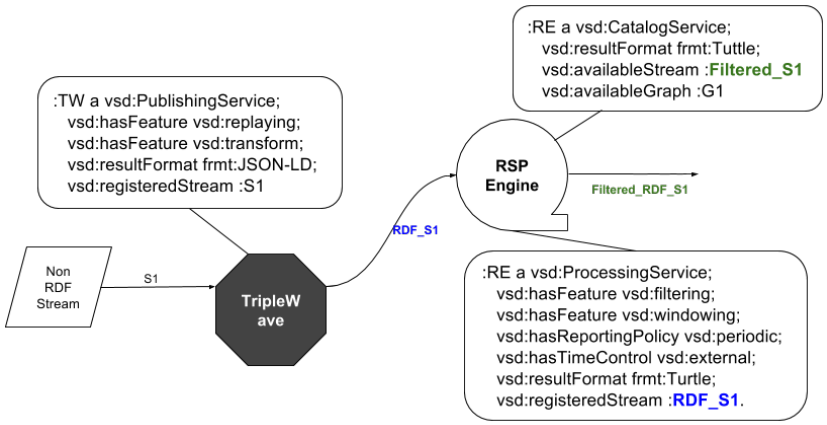

Fig. 3. Describing RDF streaming services using VoCaLS Service Description

VoCaLS Service Description makes streaming services annotation straightforward. For instance, Listing 1.5 shows an example of Service Description that uses VoCaLS to describe an instance of the C-SPARQL engine. The running

```
SELECT ?sv
WHERE {
 ?sv vsd:hasFeature vsd:filtering  ;
 vsd:registeredStream :RDF_S1 .
}
```

Listing 1.4. SPARQL query to retrieve service having vsd:filtering capabilities and stream :RDF_S1 registered

```
:csparql a vsd:ProcessingService ;
    vsd:hasFeature vsd:windowing, :timestamp_function;
    vsd:availableGraphs [ a vsd:TimeVaryingGraph ] ;
    vsd:hasRegisteredStreams [ a vocals:RDFStream ] .
:timestamp_function a vsd:RDFStreamingRSFeature ;
    dcterms:description "Takes an RDF Triple and returns its timestamp."
vsd:windowing a vsd:RDFStreamingFeature .
```

Listing 1.5. RSP Engine description using VoCaLS. Prefixes have been omitted.

engine can perform *vsd:windowing* and *vsd:filtering*, and it currently registered one RDF Stream. Another example, the RSP engine in Figure 3 has RDF_S1 stream registered and can perform filtering and windowing operations. By using VoCaLS Service Description, these services and their features can be published on the web (R2), thus allowing consumers to access service descriptions and select services suitable for their needs, as in the query in Listing 1.4 Moreover, VoCaLS is extensible and, thus, service-specific extensions are possible, e.g., in Listing 1.5, line 7 a custom feature describes a C-SPARQL timestamp function.

5.3 Stream Transformation Provenance

VoCaLS Provenance module focuses on tracking the provenance of stream processing services, i.e., tracing the consequences of operations performed over the streams. The module defines four main classes: (i) `vprov:R2ROperator` refers to operators that produce RDF mappings (relations) from other RDF mappings [2], e.g., sum and count. (ii) `vprov:R2SOperator` represents operators that produce a stream from a relation [2], for instance replaying a static dataset as a stream. (iii) `vprov:S2ROperator` refers to operators that produce relations from streams, e.g., windowing.

Finally, (iv) `vprov:S2SOperator` allows describing operators that produce a stream from another stream. To represent the most common operators, VoCaLS Provenance already contains several subclasses of the four generic ones presented above, e.g., `vprov:WidowOperator`, or `vprov:FilterOperator`.

```
SELECT ?res ?op
WHERE { ?op a vprov:Operator; prov:uses ?res.
          ?str prov:wasGeneratedBy :traffic_stream }
```

Listing 1.6. SPARQL query sources and operations generating a stream.

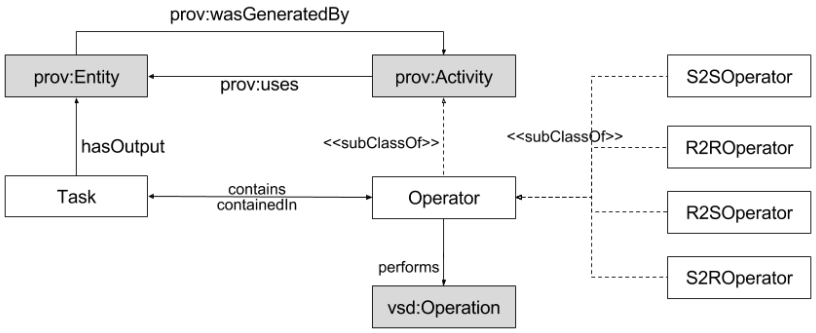

Fig. 4. VoCaLS Provenance subset for RDF stream processing operators.

Through VoCaLS Provenance, the provenance of a stream can be modeled (R4), enabling queries that indicate resources contributed to the stream or which operations were executed, as shown in Listing 1.6.

Listing 1.7 illustrates how to track the provenance of a stream that was generated as the result of `vprov:Task` which contains a sequence of `vprov:Operators`. The execution order of a set of operators is represented in a linked list by `vprov:followedBy` and `vprov:precededBy`

```
:t1 a vprov:Task; vprov:contains :op1,:op2,op3 ; vprov:hasOutput :out.
:op1 a vprov:S2ROperator; prov:uses :in_stream.
:op2 a vprov:R2ROperator; prov:uses :w1; vprov:precededBy :op1.
:op3 a vprov:R2SOperator; prov:uses :r1; vprov:precededBy :op2.
:w1 a vsd:Window; prov:wasGeneratedBy :op1.
:r1 a vocals:FiniteStatePartition; prov:wasGeneratedBy :op2.
:out a vocals:RDFStream; prov:wasGeneratedBy :op3.
```

Listing 1.7. Example of RSP operations using VoCaLS Provenance.

5.4 Combining VoCaLS with Other Vocabularies

In the following, we show how VoCaLS can be combined with existing vocabularies (R6). This is especially useful to describe the stream content. In fact, VoCaLS only focus is annotating the stream and streaming services metadata, rather then modeling the annotations within the streams. On the other hand, other ontologies such as SSN [11] and SAO [18] are important resources to describe what is streaming on. In Listing 1.8, we used these to vocabularies to enrich the stream description of Listing 1.1. The SSN ontology is used to represent the source device, and the observation data (event:Event) it produces. The SAO ontology is used to characterize information about the output of a stream observation (Stream Event).

```
:CadornaTrafficStream a ssn:Output, vocals:Stream .
:TrafficFlowSensing a ssn:sensing, sao:StreamEvent ;
  prov:used :CadornaTrafficFlow ;
```

```
  ssn:hasOutput :CadornaTrafficStream.
:CadornaTrafficSensor a ssn:SensingDevice ;
  ssn:observes  :TrafficFlow  ;
  ssn:implements :TrafficFlowSensing .
:CadornaTrafficFlow a ssn:ObservationValue, sao:StreamData ;
  prov:wasDerivedFrom :CTObservation .
:CTObservation a ssn:Observation, vsd:TimeVaryingGraph, event:Event ;
  ssn:observedProperty :TrafficFlow ;
  ssn:observationResult :CTSensorOutput ;
  event:time [a time:Instant ;
              time:inXSDDateTime "2013-01-01T00:00:00"^^xsd:dateTime ] .
```

Listing 1.8. VoCaLS with SAO and SSN Ontologies. Prefixes omitted.

6 Related Work

In this section, we position VoCaLS within the state of the art. We discuss how existing solutions already addressed some of the challenges that we presented in Section 3. Moreover, we use our requirement analysis to highlight differences and commonalities between them. Table 2 summarizes the comparison.

Linked Stream Data (LSD) [24]. This work proposed a mechanism to identify and access data streams coming from sensor networks. LSD takes into account temporal and spatial aspects and it enables discovery using URIs that models sensors, time, space and their combinations. Authors did not propose any protocol extensions w.r.t. Linked Data. Therefore, LSD supports access through streaming protocols, i.e., active paradigm. The problem of storing relevant portions of the stream is not discussed, while provenance tracking is possible for data sources using descriptions but not for stream publishing or transformations.

Requirements	R1	R2	R3	R4.a	R4.b	R4.c	R5	R6
LSD	✓	≃		✓				✓
SLD	✓		✓	✓			✓	
LDN					≃	≃	≃	≃
SAO	≃		≃	≃	✓			✓
CES		✓				✓	✓	✓
VoCaLS	✓	✓	✓	✓	✓	✓	✓	✓

Table 2. Requirements vs State-of-the-Art. Symbol Legend: Empty cell, i.e. not covered; ≃, i.e. partially covered; ✓, i.e. covered.

Streaming Linked Data (SLD) [6]. Barbieri et al. proposed SLD to publish Streaming Linked Data using an RSP engine. The approach is based on two concepts: the Stream Graph (or S-Graph) and Instantaneous Graph (or I-Graph). The S-Graph is a document that refers to stream elements and contains other relevant information. Indeed, it enables stream discovery and can partially track the provenance of the RSP engine activity. The I-Graph identifies an element of the stream. Although SLD does not propose any protocol extension, the RSP engine privileges the active paradigm to publish the stream. Nevertheless, they do not discuss how to dump stream portions nor how to track the provenance of the transformation.

Linked Data Notifications (LDN) [10] is a W3C recommendation[8] that aims at making Web Notifications de-referenceable, persistent, and reusable, i.e., compliant with Linked Data principles. Such a protocol orchestrates the communication between *senders*, *receivers* and *consumers*. Accessing the stream contents might be possible using LDN since it is not bounded to any specific transmission protocols, although the communication methods between consumer and receiver are RESTful. LDN enables the tracking of the provenance of the involved actors, but not specifically stream transformations. Although a first attempt to specialize LDN for RDF streams was presented in [9] neither discovery of streams and services nor stream finite portions are within the scope of the work, which targets communication/sharing between actors rather than exploration and querying.

Stream Annotation Ontology (SAO) [18]. SAO can be used to express the features of stream elements, i.e., *StreamEvent*s, but not Streaming services. SAO allows publishing derived data about IoT streams, and it deals with representation of aggregated data. However, the vocabulary does not aim at describing transformations in depth. Provenance tracking is possible to some extent for temporal relations. SAO can be combined with other vocabularies to enrich the description of sensor data.

Complex Event Services (CES)[15]. Gao et al. proposed CES to support smart cities applications' development using semantic technologies. The ontology extends OWL-S to support automated discovery and integration of sensor streams. It was designed to describe event services and requests, therefore it can be used to annotate streaming services. However, there is no distinction between streams publisher and consumers. Provenance tracking is possible at the level of transformation by distinguishing primitive and complex event services. The ontology is designed to be used in combination with the ACEIS middleware and other vocabularies (SSN).

7 Conclusion

In this paper, we introduced VoCaLS, a vocabulary for describing RDF streams, streaming services, and stream transformations. VoCaLS is designed to allow the publication, discovery, consumption, and provision of RDF streams. It includes the capability of describing streaming services; the operations and features that they support, and the workflows that detail how streaming data is generated or processed.

The design of VoCaLS has followed a community-driven approach, starting from the W3C RSP Community group results, and a requirement analysis described in Section 3 The challenges presented in this work and the potential impact of this ontology have been supported by the survey described in the paper, which added to the analysis of the state of the art, show the timeliness and adequacy of VoCaLS. The proposed ontology has been designed as a generic resource, which can be combined with domain-specific vocabularies (e.g. for IoT),

[8] `https://linkedresearch.org/ldn/`

and reuses and inherits elements from widely used vocabularies such as DCAT and VoID. Furthermore, VoCaLS is also the result of the evolution and merge of two independent preliminary works [23,14] in this area.

VoCaLS has been published following well-principled practices for the publication of the vocabulary, including the set up of permanent URIs, the availability of full open documentation using Widoco, the availability of sources in Github[9], its inclusion in the LOV repository, and the setup of redirects for serving different ontology formats.

Road Map: Regarding the adoption and sustainability plans for VoCaLS, several steps have been taken in this direction. First, given that the establishment of a common vocabulary is one of the main goals of the W3C RSP Community Group, we have started the process of elevating this vocabulary as an official Group Note. The adoption and support from the authors, as a relevant part of this community, will contribute positively to this endeavor. Once this step is achieved, the RSP Community Group is expected to take the responsibility for maintenance, updates and dissemination.

Second, an important goal is to foster the adoption of VoCaLS within relevant communities. For this purpose we initiated the creation of a catalog of streams descriptions[10]. We started annotating all the historical streams that where published for benchmarking purposes. Moreover, we developed a simple utility[11] to support the annotation of new streams. Finally, in order to lead by example, we have launched the integration of VoCaLS within relevant services and software available for the RSP community: the RSP Services, RSPLab, and TripleWave.

References

1. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets. In: Workshop on Linked Data on the Web, LDOW (2009)
2. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. VLDB J. 15(2), 121–142 (2006)
3. Balduini, M., Della Valle, E.: A restful interface for RDF stream processors. In: ISWC (Posters & Demos). vol. 1035, pp. 209–212. CEUR-WS.org (2013)
4. Balduini, M., Della Valle, E., Dell'Aglio, D., Tsytsarau, M., Palpanas, T., Confalonieri, C.: Social listening of city scale events using the streaming linked data framework. In: ISWC. vol. 8219, pp. 1–16 (2013)
5. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-sparql: a continuous query language for rdf data streams. Intl. J. Semantic Computing 4(01), 3–25 (2010)
6. Barbieri, D.F., Della Valle, E.: A proposal for publishing data streams as linked data - A position paper. In: LDOW (2010)
7. Berners-Lee, T., Bizer, C., Heath, T.: Linked data-the story so far. IJSWIS 5(3), 1–22 (2009)

[9] `https://github.com/ysedira/vocals`
[10] `https://github.com/ysedira/vocals/tree/master/catalog`
[11] `https://github.com/ysedira/stream-annotation-tool`

8. Breslin, J.G., Decker, S., Harth, A., Bojars, U.: Sioc: an approach to connect web-based communities. Intl J of Web Based Communities 2(2), 133–142 (2006)
9. Calbimonte, J.P.: Linked data notifications for rdf streams. In: Proc. of the Web Stream Processing (WSP) Workshop at ISWC. pp. 66–73 (2017)
10. Capadisli, S., Guy, A., Lange, C., Auer, S., Sambra, A.V., Berners-Lee, T.: Linked data notifications: A resource-centric communication protocol. In: ESWC (1). vol. 10249, pp. 537–553 (2017)
11. Compton, M., Barnaghi, P., Bermudez, L., GarcíA-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., et al.: The ssn ontology of the w3c semantic sensor network incubator group. J Web semantics 17, 25–32 (2012)
12. Della Valle, E., Ceri, S., Van Harmelen, F., Fensel, D.: It's a streaming world! reasoning upon rapidly changing information. IEEE Intelligent Syst. (6), 83–89 (2009)
13. Dell'Aglio, D., Della Valle, E., Calbimonte, J.P., Corcho, O.: Rsp-ql semantics: a unifying query model to explain heterogeneity of rdf stream processing systems. Intl. J. on Semantic Web and Information Systems (IJSWIS) 10(4), 17–44 (2014)
14. Dell'Aglio, D., Le Phuoc, D., Le-Tuan, A., Ali, M.I., Calbimonte, J.P.: On a web of data streams. In: ISWC DeSemWeb (2017)
15. Gao, F., Ali, M.I., Curry, E., Mileo, A.: Automated discovery and integration of semantic urban data streams: The ACEIS middleware. Future Generation Comp. Syst. 76, 561–581 (2017)
16. Gao, S., Scharrenbach, T., Bernstein, A.: The clock data-aware eviction approach: Towards processing linked data streams with limited resources. In: European Semantic Web Conference. pp. 6–20. Springer (2014)
17. Garijo, D.: Widoco: a wizard for documenting ontologies. In: International Semantic Web Conference. pp. 94–102. Springer (2017)
18. Kolozali, S., Bermúdez-Edo, M., Puschmann, D., Ganz, F., Barnaghi, P.M.: A knowledge-based approach for real-time iot data stream annotation and processing. In: 2014 IEEE International Conference on Internet of Things, Taipei, Taiwan, September 1-3, 2014. pp. 215–222 (2014)
19. Le-Phuoc, D., Dao-Tran, M., Parreira, J.X., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: ISWC, pp. 370–388 (2011)
20. Le-Phuoc, D., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. J. Web Semantics 16, 42–51 (2012)
21. Maali, F., Cyganiak, R., Peristeras, V.: Enabling interoperability of government data catalogues. In: Electronic Government, 9th IFIP WG 8.5 International Conference, EGOV 2010. pp. 339–350 (2010)
22. Mauri, A., Calbimonte, J.P., Dell'Aglio, D., Balduini, M., Brambilla, M., Valle, E.D., Aberer, K.: TripleWave: Spreading RDF Streams on the Web. In: ISWC. pp. 140–149 (2016)
23. Sedira, Y.A., Tommasini, R., Della Valle, E.: Towards vois: a vocabulary of interlinked streams. In: ISWC DeSemWeb (2017)
24. Sequeda, J.F., Corcho, O.: Linked stream data: A position paper. In: SSN. pp. 148–157. CEUR-WS. org (2009)
25. Tommasini, R., Della Valle, E., Mauri, A., Brambilla, M.: Rsplab: RDF stream processing benchmarking made easy. In: ISWC. pp. 202–209 (2017)
26. Vandenbussche, P.Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked open vocabularies (lov): a gateway to reusable semantic vocabularies on the web. Semantic Web 8(3), 437–452 (2017)