Semantic Containers for Data Mobility: A Seismic Activity Use Case*

Fajar J. Ekaputra[1], Peb R. Aryan[1], Elmar Kiesling[1], Christoph Fabianek[2], and Eduard Gringinger[2]

[1] TU Wien, Vienna - Austria
{fajar.ekaputra,peb.aryan,elmar.kiesling}@tuwien.ac.at
[2] OwnYourData, Bad Vöslau - Austria
link {christoph,eduard}@ownyourdata.eu

Abstract. An enormous wealth of data is being created in our increasingly digitized life and economies, but the majority of this data is either being monopolized or shared for questionable purposes rather than facilitating innovation. One of the main issues that inhibits data reuse and the emergence of vibrant data markets is the lack of standard mechanisms for the execution of controlled transactions between data providers and consumers, including efficient provisioning of data and associated services. Consequently, many potential data providers find it difficult to turn data into viable business models. At the same time, data consumers face challenges accessing heterogeneous data provided via various mechanisms and have limited means to trace data provenance and ensure data quality. Semantic Containers aim to tackle both issues by facilitating controlled transactions through an integrated set of methods and capabilities. Specifically, they package data, semantic descriptions, and processing capabilities into executable and shareable containers. In this paper, we illustrate the concept by means of a pilot project in collaboration with the largest meteorological institute in Austria that illustrates how Semantic Containers can be used to provide Seismic activity data.

Keywords: Data Mobility · Container · Semantic Technologies

1 Introduction

The vast economic potential of data as a cornerstone of modern economies has grown tremendously in many domains in recent years. This is partly driven by technological advances in big data analytics and machine learning and also reflected in the fast growth of data, which currently amounts to an estimated of 2.5 quintillion bytes of data being generated per day[1]. Despite open data initiatives and some efforts to create markets for data in particular domains, however, data is typically not commonly shared and reused. Moreover, potential data

* This work was funded by the Austrian Research Promotion Agency FFG under grant 869781 (SEMCON)

[1] https://rebrand.ly/data-generation-per-day

consumers continue to face tedious and time-consuming hurdles when buying, accessing or working with data from external sources.

This may be attributed partly to technical issues and partly to considerations and questions over the viability of business models built around the sales of data. From a data provider perspective, providing access to data often implies giving up control over the terms under which the data will be used. From a data consumer's perspective, both technical challenges and limited trust in the quality, completeness, and provenance of available data have contributed to the limited adoption of data markets today. This is unfortunate, given that data can be replicated and moved at very low cost and can generate great economic value when being shared.

Semantic Containers aim to address these challenges by providing a lightweight, efficient, transparent, and standardized infrastructure for data provisioning between involved parties. Specifically, we developed a framework to associate data with sticky policies, processing services, and provisioning mechanisms that can be transferred and provisioned easily in distributed settings. The proposed concept allows data providers to distribute data while retaining a level of control over its usage, while at the same time providing data consumers well-managed mechanisms to obtain and integrate data in a standardized manner. To this end, we package data and processing capabilities into reusable containers, describe their semantics and permissible usage, and provide uniform interfaces for access and reuse. Thereby, a data set becomes a commodity with well-defined content, properties and usage policy, as well as clear usage rights.

This demo paper presents a pilot prototype of the Semantic Container implementation in a project developed in collaboration with the largest meteorological institute in Austria. In the remainder of the paper, we first briefly introduce the Semantic Container concept in Section 2 and follow it up with the pilot implementation and application in the context of seismic data in Section 3. Finally, we conclude with an outline of future work directions in Section 4.

2 Semantic Containers

Semantic Containers are designed to support and facilitate data exchange between multiple parties. Essentially, each Semantic Container (SemCon) provides the ability to *(i)* **accept input data**, *(ii)* **validate and provision data** with necessary information, e.g., usage policy and provenance, *(iii)* **record data hashes** in a distributed ledger for immutability, *(iv)* **store data**, and *(v)* **distribute output data to users**. In addition, SemCon provides an optional *(vi)* **billing service mechanism** to facilitate transaction over data exchange.

Each SemCon describes its content using semantic web standards and vocabularies. Furthermore, semantic descriptions are also used for SemCon services, e.g., for data validation, usage policy description, and data provenance representation. The use of semantic descriptions facilitates interoperability among SemCons, which in turns allows users to develop data processing pipelines out of them. SemCons can be customized further for particular use cases. For exam-

ple, the seismic activity use case covered in this paper proposes a container that provides query capabilities in addition to basic data distribution.

Our approach can be characterised as follows: *(i)* **Provisioned**: It provides a well-defined, reproducible, and automatically verifiable data state by keeping data hash records in a distributed ledger. *(ii)* **Distributed**: It supports data exchange between two or more users without the need for an intermediary. *(iii)* **Open Source**: It is available to the general public for use or modification from its original design. *(iv)* **Interoperable**: Two or more of SemCons can be combined to create a data processing pipeline. *(v)* **Packaged**: It combines data, semantic description, and program logic in a single distribution mechanism. *(vi)* **Policy-tracking**: It provides a mechanism to keep track of permissible usage based on license terms and user consent for the processing of personal data.

We use a broad set of technologies to implement the framework. Docker[2] serves as a foundation for the containerization. We documented the SemCon API descriptions using Swagger[3]. The hash value of a record (containing data, associated usage policy, and provenance trail) is stored in a distributed ledger to prove immutability. The default way to store these hash values in SemCon is using a notary service[4] based on the public blockchain Ethereum[5]. For the independent verification of hash value and blockchain address an audit proof is also provided by the notary service according to RFC 6962. For the representation of the semantic descriptions of data models and for configuration, we use an RDF-based format. Furthermore, we use SHACL constraints [3] for data and configuration validation. Finally, we adopt the SPECIAL vocabulary [1] to describe the usage policy of the data and PROV-O [4] for provenance information. The mechanism used for checking compliance with permissible usage policies is adapted from our previous work [2].

The software for handling API requests and response is primarily written in Ruby. The validation and processing services for semantic descriptions are written in Java, using Apache Jena[6] for RDF data management and the caRML library[7] for data acquisition. All resources are available on our GitHub repository[8] and on DockerHub[9] under an MIT license. We also provide a basic tutorial on how to use SemCon on our GitHub readme file[10].

3 Seismic Data Use Case

Based on the design defined in the previous section, we piloted a prototype partnering with the largest meteorological institute in Austria to provide data

[2] `https://www.docker.com`
[3] `https://api-docs.ownyourdata.eu/semcon/`
[4] `https://notary.ownyourdata.eu`
[5] `https://www.ethereum.org`
[6] `https://jena.apache.org`
[7] `https://github.com/carml/carml`
[8] `https://github.com/sem-con`
[9] `https://hub.docker.com/u/semcon`
[10] `https://github.com/sem-con/Tutorials`

on seismic events, such as earthquakes. Prior to our implementation, our use case partner already had an API services for distributing seismic data from their database in JSON format[11]. The main goal of the pilot project is to demonstrate how Semantic Containers could extend the data provision mechanism for data providers and improve data access for data consumers.

```
@prefix : <http://w3id.org/semcon/ns/ontology#>. # other prefixes omitted
# Graph :BaseConfiguration - basic container information
:BaseConfiguration { # datatype properties are omitted
  :ContainerConfigurationInstance rdf:type :ContainerConfiguration ;
    :hasDataConfiguration :DataConfigurationInstance .
  :DataConfigurationInstance rdf:type :DataConfiguration ;
    :hasNativeSyntax <http://w3id.org/semcon/ns/ontology#JSON> }
# Graph :UsagePolicies - (mandatory) permissible usage of data
:UsagePolicy { } # Contents omitted
# Graph :DataModel ; (optional) Ontology structure of data
:DataModel { } # Contents omitted
# Graph :DataConstraint - (optional) SHACL constraints for data validation
:DataConstraint { } # Contents omitted
# Graph :DataMapping - (optional) RML Mapping for data transformations
:DataMapping { } # Contents omitted
# Graph :UserFunction - (future) semantic descriptions for added functions
:UserFunction { } # Contents omitted constraints
```

Listing 1: An excerpt semantic descriptions (`init.trig`) of a `sc-sparql`[12]

To achieve these goals, we developed two containers for this seismic use case: *(i)* `sc-seismic`[13] to handle interoperability with the partner's original API, and *(ii)* `sc-sparql`[14] to transform output data from `sc-seismic` into RDF and provide users with a SPARQL endpoint for the seismic data. Both containers extend the `sc-base`[15] container, which provides basic functionalities for all containers, including configuration mechanisms and an API structure.

The **sc-seismic** container wraps the original API access and associates it with additional information (i.e., sticky permissible usage policy). The technical details of accessing data from original API is abstracted out into a standard API call of the SemCon. Each API access to the SemCon is also timestamped and hashed in order to track each access. Therefore, SemCon reduces the workload of the data provider, who can focus more on providing high-quality data.

The **sc-sparql** container extends `sc-base` with a SPARQL endpoint to allow data consumers to access seismic data with SPARQL queries. This container

[11] `https://geoweb.zamg.ac.at/static/event/lastday.json`
[12] The complete init.trig is available at `https://rebrand.ly/sc-sparql-init-trig`
[13] `https://github.com/sem-con/sc-seismic`
[14] `https://github.com/sem-con/sc-sparql`
[15] `https://github.com/sem-con/sc-base`

provides RDF graph data constructed from the original data. In order to achieve this, the container is equipped with an RDF mapper, an RDF data validator, and a SPARQL query service that uses the semantics described in the `init.trig` configuration file (cf. Listing 1 for an example). First, an embedded RDF mapper will transform the data using an RDF mapping specification in RML[16] (Line 16) to augment the data with semantic models. Second, to ensure data quality, an RDF data validation service will check the generated RDF using specified SHACL constraints (Line 14). Third, the resulting RDF graph will be stored in the `sc-sparql` triplestore, ready to be queried. Furthermore, we ensure that the sticky policy of the `sc-sparql` (Line 10) is compatible with the source container (i.e., `sc-seismic`). In summary, with `sc-sparql`, users are able to query the RDF Graph using standardized query mechanism and generation process while keeping track of the policy and generating a provenance chain.

The above containers separate the functionality to enable scaling as each of the container can be deployed independently in a distributed setting (e.g., during high load or complex queries, the deployment of `sc-sparql` can be scaled up while `sc-seismic` may remain as the only container responsible for 'raw' data provision). Combined, the above containers work as a pipeline to expose high-quality data through a SPARQL endpoint. A graphical user interface to interact with a `sc-sparql` that is connected to a `sc-seismic` is available online[17].

4 Conclusion and Future Work

In this paper, we have introduced the Semantic Container concept and its pilot prototype in the context of seismic event data. With our demo, we showcase our approach and gather valuable feedback from the community for further development and evaluation. We are presently working to extend the concept with data monetization for `sc-seismic` as well as prototype development for two additional use cases, i.e., *(i)* satellite observation data to demonstrate the ability of SemCon in handling large amount of data, and *(ii)* diabetes patients data collections to exhibit the capability of SemCon in handling sensitive data.

References

1. Fernández, J., Bonatti, P., Kirrane, S., Petrova, I.M., Sauro, L., Schlehahn, E.: The **SPECIAL** usage policy language, unofficial draft 06 april 2018. URL: https://aic.ai.wu.ac.at/qadlod/policyLanguage/ (2018)
2. Fernandez Garcia, J.D., Ekaputra, F.J., Aryan, P.R., Azzam, A., Kiesling, E.: Privacy-aware **Linked Widgets** (2019)
3. Knublauch, H., Kontokostas, D.: Shapes constraint language (**SHACL**), **W3C** recommendation 20 july 2017. URL: https://www.w3.org/TR/shacl (2017)
4. Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: **PROV-O**: The **PROV** ontology, **W3C** recommendation 30 april 2013. URL: https://www.w3.org/TR/prov-o/ (2013)

[16] `https://rml.io`

[17] `https://rebrand.ly/sc-seismic-sparql`; `https://rebrand.ly/sc-gui-readme`