OGC® DOCUMENT: 21-054

External identifier of this OGC® document: http://www.opengis.net/doc/PER/DP21-JSON-LD

Open Geospatial Consortium

OGC DISASTER PILOT JSON-LD STRUCTURED DATA ENGINEERING REPORT

ENGINEERING REPORT

PUBLISHED

Submission Date: 2022-06-28
Approval Date: 2022-08-16
Publication Date: 2023-01-05
Editor: Sergio Taleisnik

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

I EXECUTIVE SUMMARY

The key to disaster awareness and better decision making in emergency situations is making the right information available to the right people at the right time. This is a particularly acute challenge for the affected public who may be limited or, conversely, overwhelmed in their access to information. Often the population in areas impacted by disaster rely on Web searches to find the information needed to make the right choices at the right time. There is considerable promise, then, in optimizing the ability of search engines to present and prioritize such information, often by providing additional web page header content in the form of linked, structured data that enables the search results to provide direct answers to important questions.

As part of the Disaster Pilot 2021, experiments were conducted in publishing web pages containing this additional content to see whether searches could be optimized to provide more actionable disaster information. The structured data content was further augmented with resolvable links containing globally unique identifiers for facilities such as clinics and hospitals, making it more likely that search queries could unambiguously identify those facilities.

The Pilot activities related to linked structured data provided the following four challenges and lessons learned.

- **Challenge 1**: OGC API — Features proved to be very powerful in providing access to geographic data. The use of templates was deemed necessary to speed up the process of generating both the JSON-LD and the HTML out of a large number of features.
- **Challenge 2**: Mapping some of the terms has been more challenging than expected for two main reasons. First, vocabulary types loosely overlap with actual datasets, or are too generic. Second, the structure of the vocabulary types makes it hard to map statistics and geographical locations in a same feature.
- **Challenge 3**: For indexing, Google accepts only HTML documents identified by a URL without any additional parameter, limiting the usage of dynamic pages and filters.
- **Challenge 4**: Data needs to be included in both the HTML body and the JSON-LD `@context`. Since the latter is actually embedded into the HTML, the data ends up being duplicated. This can potentially affect the loading speed of the webpages.

Publishing the HTML Feature output posed an additional problem: For indexing, Google accepts for only HTML documents identified by a URL without any additional parameter. However, any `GetFeature/GetFeatureInfo` operation performed on a GeoServer instance must include within the URL several query parameters such as the service, the operation, the `featureType` being requested, and the output format, to cite few. This forced GeoSolutions to create and publish a static HTML page with the encoded `FeatureCollection` and the JSON-LD document.

Four main recommendations emerged from the Pilot activities related to Linked Data.

- **Rec 1**: Currently, JSON-LD output format is supported only when retrieving the data (the collection items). By expanding this support to the retrieval of metadata, it could be then possible to improve the indexability of GIS websites.

- **Rec 2**: To prevent having the same data in both the JSON-LD `@context` and in the HTML body, using linked references to the JSON-LD `@context` would at least separate it from the HTML, speeding the retrieval of both.
- **Rec 3**: Having vocabulary types with geospatial properties appear as instances for properties of other vocabulary terms would make it easier to map vocabulary types to disaster-related datasets, which usually include a geospatial property.
- **Rec 4**: The use of GLNs with Linked Data could boost the discoverability of disaster-related information by enabling the search of such information using GLNs. This type of search could return disaster-related information associated with a specific area or facility identified by a GLN.

II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, JSON-LD

III SECURITY CONSIDERATIONS

No security considerations have been made for this document.

IV SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Skymantics

V DOCUMENT CONTRIBUTOR CONTACT POINTS

All questions regarding this document should be directed to the editor or the contributors:

Table — Document Contributor Contact Points

NAME	ORGANIZATION	ROLE
Sergio Taleisnik	Skymantics	Editor
Nuno Olivera	GeoSolutions	Contributor
Marco Volpini	GeoSolutions	Contributor
Phil Archer	GS1	Contributor
Josh Lieberman	OGC	Contributor

VI FOREWORD

Some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

1 SCOPE

1 SCOPE

This report describes an experiment designed to understand how search engines can use linked data content to generate search results that are more immediately useful in emergency situations. Disaster information was published using web pages containing structured data and uniquely resolvable facility identifiers in their headers.

2

SUBJECT

2 SUBJECT

This Disaster Pilot JSON-LD Structured Data Engineering Report documents the analysis, discussions, results, and recommendations that emerge from the efforts carried out regarding the use of JSON-LD with OGC APIs to generate structured web page data for search engine optimization of disaster related information.

This ER provides the practical experience and lessons learned on the usage of Linked Data within OGC APIs with the objective of enhancing the web search and finding up-to-date conditions, observations, and predictions associated with well-known local geography. Upcoming initiatives should use the findings documented in this ER to further develop applications that make geospatial data and information more easily findable, accessible, interoperable, and reusable, which will increase the efficiency of disaster response. This ER could also be used as a case study of Linked Data to help other industries understand its value and implement it within their domains, or it could serve as a baseline for adding Linked Data support to one or several OGC API standards.

3

TERMS, DEFINITIONS, AND ABBREVIATED TERMS

3 TERMS, DEFINITIONS, AND ABBREVIATED TERMS

3.1. Terms and Definitions

3.1.1. Linked Data

Structured data which is interlinked with other data.

3.1.2. JSON-LD

JavaScript Object Notation for Linked Data (JSON-LD) is a method of encoding linked data using JSON.

3.1.3. Ontology

There is no clear division between what is referred to as “vocabularies” and “ontologies.” The trend is to use the word “ontology” for more complex and formal collections of terms, whereas “vocabulary” is used when such strict formality is not necessary. [6]

3.1.4. Semantic Web

The term “Semantic Web” refers to W3C’s vision of the Web of Linked Data. Semantic Web technologies enable the creation of data stores on the Web, the building of vocabularies, and the writing of rules for handling data. [6]

3.1.5. Vocabulary

On the Semantic Web, vocabularies define the concepts and relationships (also referred to as “terms”) used to describe and represent an area of concern. Vocabularies are used to classify the terms that can be used in a particular application, characterize possible relationships, and define possible constraints on using those terms. [6]

3.1.6. Web GIS

An advanced form of Geographic Information Systems available on web platforms that could make use of Semantic Web Technologies.

3.2. Abbreviated terms

API	Application Programming Interface
ER	Engineering Report
GIS	Geographic Information System
GLN	Global Location Number
JSON	JavaScript Object Notation
OGC	Open Geospatial Consortium
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

4

INTRODUCTION

4 INTRODUCTION

A disaster, whether anticipated or not, can be an overwhelming event. Each disaster has its own unique details and progression, which can generate and cascade into additional crisis situations. The preparation and coordination of scalable responses can mitigate these challenges. Disaster Pilot 2021 brings the technological pieces together in order to reduce information preparation time and accelerate the ability to transform data from observation into decision. This will require bridging the divides between providers, responders, and other stakeholders, forming a connected ecosystem of data and technologies and developing the capacity to produce Decision Ready information products that answer decision makers' questions almost as fast as they can be posed.

Such information products are currently not as useful as they could be if they can't be accessed and used by stakeholders, especially the affected public. Web publication of "structured data" that links well-known content such as local geography with up-to-the-minute situation details enables search engines to push disaster-relevant information up in search results and help the public to stay on top of fast-moving events. This should be possible through the proper combination of technologies, geospatial standards, and data sharing arrangements that bring the right information at the right time to the right people in the right place.

This report specifically covers experiments relating to optimization of Web search results for actionable disaster related information. It details work adding "structured data" to disaster information published on the Web and demonstrating how the OGC-API suite can be integrated with JSON-LD and aligned with schema.org. This will enhance the semantic definition of the OGC- API schemas and give a clear path for OGC-API generated content to be incorporated into web search engine indexes, with the goal of pushing disaster-relevant information up in search results during an ongoing disaster.

5

BACKGROUND

5 BACKGROUND

5.1. Use of Linked Data in Web Search

Structured data is a standardized format for providing information about a page and classifying the page content; for example, on a recipe page, what are the ingredients, the cooking time and temperature, the calories, and so on.

Search engines, like Google, use structured data found on the web to understand the content of the page, as well as to gather information about the web and the world in general. Google Search also uses structured data to enable special search result features and enhancements that help users find the information they are looking for as well as related information.

Google provides documentation describing which properties are required, recommended, or optional for structured data. Most Search structured data uses Clause 5.2 vocabulary, but the Google Search Central documentation is the definitive for Google Search behavior, rather than the schema.org documentation. There are more attributes and objects on schema.org that aren't required by Google Search, although they may be useful for other services, tools, and platforms. [1]

5.2. Schema.org

Schema.org is a collaborative community activity with a mission to create, maintain, and promote schemas for structured data on the Internet, on web pages, in email messages, and beyond.

Schema.org vocabulary can be used with many different encodings, including RDFa, Microdata and JSON-LD. These vocabularies cover entities, relationships between entities, and actions and can easily be extended through a well-documented extension model. Over 10 million sites use Schema.org to markup their web pages and email messages. Many applications from Google, Microsoft, Pinterest, Yandex, and others already use these vocabularies to power rich, extensible experiences. [2]

For features specifically related to location, schema.org provides a schema type named **Place**, representing *Entities that have a somewhat fixed, physical extension*. The schema type includes properties related to describing the location of an entity through different methods, including a property specific for GLN

5.3. Global Location Number (GLN)

The Global Location Number (GLN) is used to identify locations and legal entities. This unique identifier comprises a GS1 Company Prefix, Location Reference, and Check Digit.

GLNs are used to identify parties to business transactions; functional groups within a company; or real, physical “places” that might ship, receive, process, or hold inventories [3]. Examples include:

- **Legal entities**: whole companies, subsidiaries or divisions within a company, health system corporations, etc.;
- **Functional entities**: specific departments within a legal entity, such as an accounting department, purchasing department, hospital pharmacy, etc.;
- **Physical locations**: manufacturing facilities, distribution centers, warehouses, dock doors, hospital wings, bin locations, retail stores, etc.; and
- **Digital locations**: an electronic or non-physical address such as an EDI gateway or ERP system.

5.4. OGC Explorations on Linked Data and Earth Observation (EO)

This ER addresses the merge of three lines of technical research, shown in Figure 1, that have been explored by OGC Pilots and Innovation Initiatives over the past years.

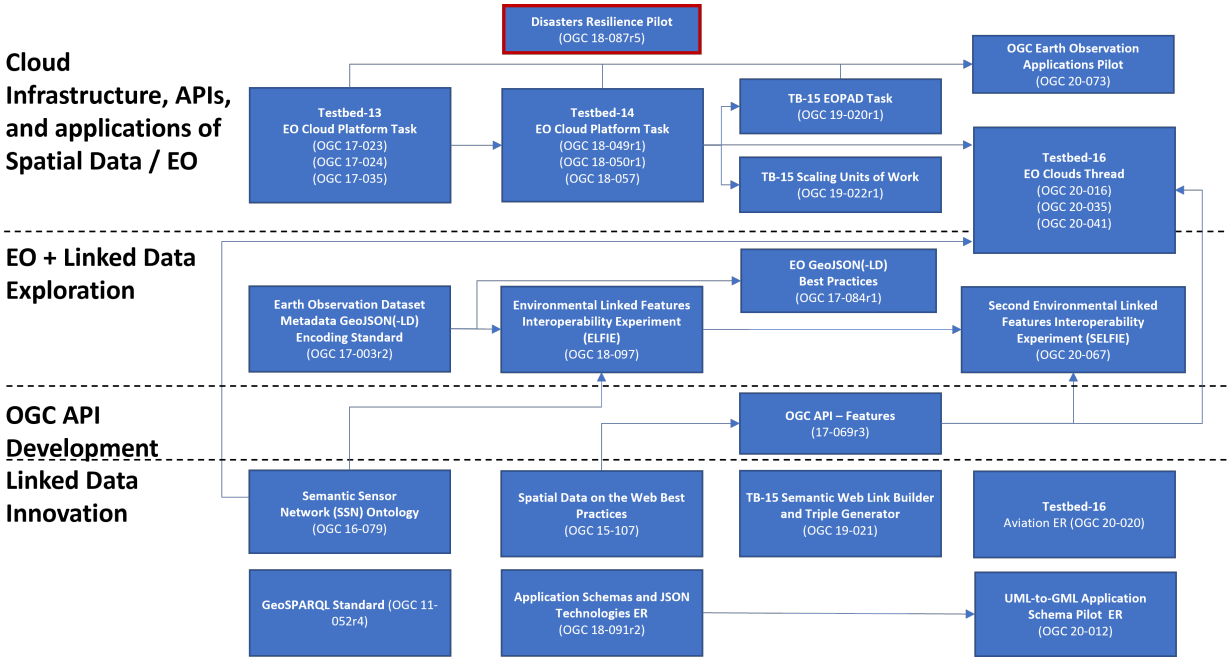

Figure 1 — OGC Explorations on Linked Data and EO

The first line of research was the development of **cloud data infrastructures and web services for the publication, usage, and consumption of raw data, analysis ready data (ARD), and decision ready information (DRI) related to earth observation (EO) and disaster response**. Testbeds 13 through 15 included a series of tasks related to the development of cloud environments to support EO applications, while work on Testbed-16 used this cloud framework together with the emerging new OGC API Standards to explore the generation of cloud-based EO Analysis Ready Data (ARD) and the development of the Data Access and Processing API (DAPA), simplifying access to environmental and earth observation data.

The OGC Disasters Resilience Pilot worked on the demonstration of the usefulness of standards and SDI architecture within the Disaster community and provided recommendations for future work. Finally, last year, the OGC Earth Observation Application Pilot also explored the architecture that was developed in OGC Testbeds 13 through 15 and developed arrangements and definitions that enabled the deployment and execution of applications on various platforms with minimal adaptations.

The second line of research was the exploration of **Linked Data applications in the EO domain**. In 2017 OGC released the Earth Observation Dataset Metadata GeoJSON(-LD) Encoding Standard (OGC 17-003r2), describing a GeoJSON and JSON-LD encoding for EO metadata for datasets. Two years later, OGC released a best practices document for EO GeoJSON and JSON-LD (OGC 17-084r1), documenting models for the exchange of information describing EO collections, both within and between different organizations.

The OGC Environmental Linked Features Interoperability Experiment (ELFIE) (OGC 18-097) began exploring the use of Linked Data and JSON-LD within the environmental domain and sampling features. Two years later, the Second ELFIE (OGC 20-067) explored the use of the OGC API — Features APIs with environmental Linked Data, expanded the use of vocabularies

from schema.org in the EO domain, identified the need for structured data encoded in JSON-LD, and focused on schema.org and other common ontologies.

Last year, for OGC Testbed-16, an API endpoint was deployed consistent with the emerging OGC API family of standards, oriented to observation data and implementing Linked Data. The API provided access to weather observations and its responses included a JSON-LD 1.1 context that related the JSON values to definitions in the W3C SSN Ontology and the GeoJSON vocabulary so that JSON-LD-aware clients could process the observation data. (OGC 20-016).

Lastly, the third line of research identified was the development of technologies that enhance the **use of Linked Data** within the community.

On 2012, OGC released the GeoSPARQL Standard (OGC 11-052r4) which defined a vocabulary for representing geospatial data in RDF (Resource Description Framework) and defined an extension to the SPARQL query language for processing geospatial data. OGC worked in partnership with W3C to generate ontologies that now have the potential to support use cases related to disaster response: In 2017 OGC 16-079 defined the Semantic Sensor Network (SSN) Ontology for describing sensors and their observations, which includes a lightweight but self-contained core ontology called SOSA (Sensor, Observation, Sample, and Actuator).

The Spatial Data on the Web Best Practices (OGC 15-107), released on 2017 by OGC and W3C, included specific recommendations to promote the discovery of web pages for spatial things in search engines, including using unique pages for each feature, enhancing the available metadata, using JSON-LD, and adding links to related features. The report provided base concepts for building the OGC API — Features Standard (17-069r3), including Linked Data principles

In 2018, OGC 18-091r2 investigated how the semantics of JSON data can be defined through the use of JSON-LD. The analysis identified a number of issues, potential solutions, recommendations, and best practices. OGC 20-012 took part of this research and investigated the implications of a few JSON Schema conversion rules regarding the ability for mapping JSON data that complies to these rules to RDF using JSON-LD and provided further lessons learned and recommendations.

6

TECHNICAL ARCHITECTURE

6 TECHNICAL ARCHITECTURE

This chapter presents the goals defined for this Pilot, the components that were built to address those goals, and the interactions among them.

6.1. Problem Statement

Disaster management frequently encounters key data sharing challenges that make present solutions more difficult, slower, and less effective than they could be for the following reasons.

- Data, particularly EO data, can be hard to find, complicated to share, difficult to access, and slow (or unable) to be processed into common forms that are suitable for analysis and integration.
- Integration of diverse data sources into end-user information products can, in turn, be both arduous and slow to adapt to the needs of particular disaster situations by users and responders.
- End-user information products in their volume and frequency often overwhelm the connectivity available to responders and relief organizations in impacted regions.
- Local information such as in situ sensor observations, field reports, and volunteered information, are often difficult to collect and even more difficult to incorporate back into provided information products.
- Up-to-date and actionable event information, even when openly available, can be difficult for the affected public to find and stay on top of. [4]

The goal of this Pilot was to explore and advance geospatial standards-based solutions for improving disaster management. The process included developing prototypical components and services that utilize modern cloud architecture and next generation technologies. This will optimize collaborative workflows that are able to rapidly provide scalable provision ARD and DRI products, services, and applications.

The activity will address the following key components of a geospatial information flow for disaster management operations.

- Reduced Time to Discover, Access and Transform Data: Near-real-time cloud-based discovery, processing, and access of analysis ready geospatial data (ARD) from diverse sources.
- Analysis and Decision Ready Services: Analysis, visualization, and collaboration processes enabling generation of situation-appropriate , decision-ready information and indicators (DRI).

- Decision Support: On-demand and event-driven dissemination of DRI to responders, decision makers, and other disaster stakeholders.
- Mobile Devices: Use of offline data containers to work with DRI in the field under connected-disconnected conditions.
- Work the Web: Enhancements to data services to optimize discoverability by mainstream search engines and potential of linked data approaches to improve public awareness. [4]

The main goals of the Disaster Pilot were: to explore and advance geospatial standards-based solutions for improving disaster management by rapid prototyping and experimentation with Web APIs; to develop GeoPackage viewers that effectively support field operations in disaster management situations; to exercise discovery, virtual collaboration, and dynamic integration of data from various sources; and to assess connected-disconnected dissemination.

An additional goal of the Pilot was to understand how structured data can help with search engine optimization [4], which is addressed in this ER. Another ER describes the activities related to the previous objectives. [5]

Within the Disaster Pilot, the specific requirement regarding the use of structured data was the demonstration of how the OGC-API suite can be integrated with JSON-LD and aligned with schema.org to enhance the semantic definition of the OGC- API schemas and give a clear path for OGC-API generated content to be incorporated into web search engine indexes [4].

6.2. Functional Overview

To fulfill its goals and requirements, this Pilot was organized into a collection of interconnected components, as seen on Figure 2. Each component addressed one or several Pilot requirements. Among these components, the **JSON-LD Structured Data Generator** (identified as *D109*) was created to address the goals related to structured data.

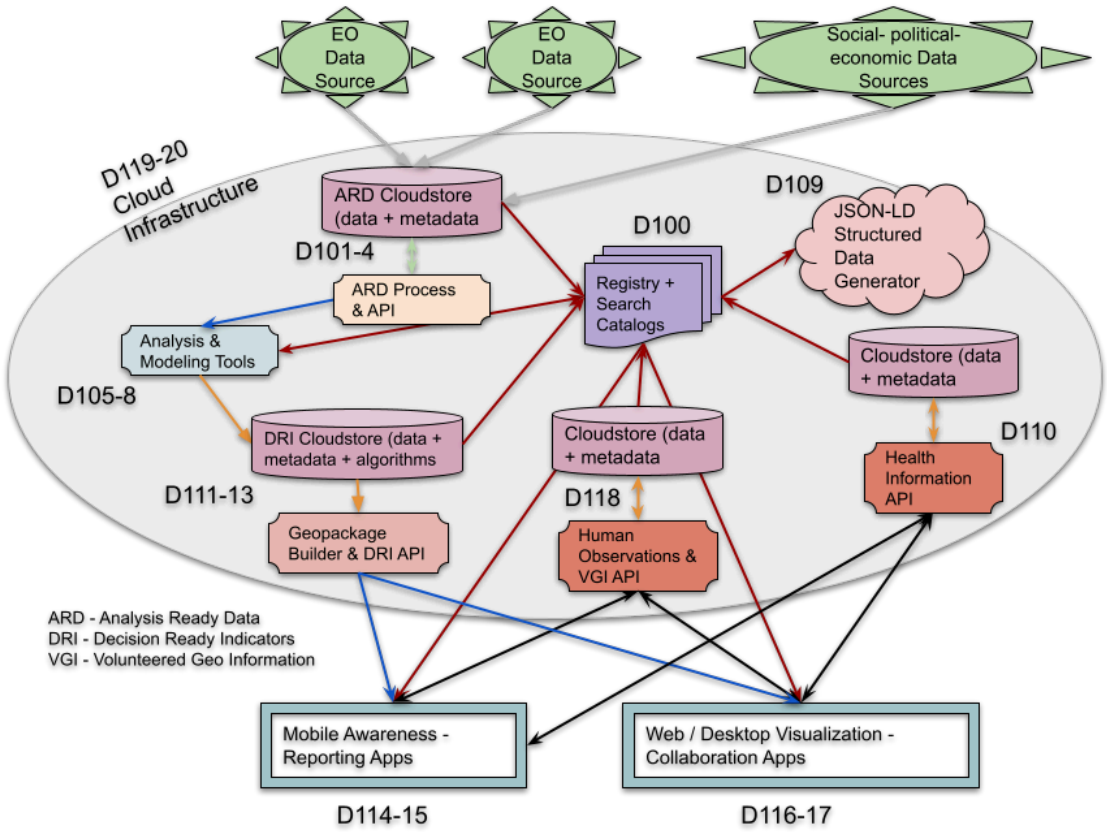

Figure 2 — Disaster Pilot Components

The JSON-LD Structured Data Generator would be designed to draw from other Pilot components to generate and publish disaster-oriented web content that incorporates structured data in order to make relevant and up-to-date decision-ready information visible more prominently and specifically in major search engine results. This would include integrating OGC API services that provide such information with JSON-LD and aligning with schema.org in order to give a clear path for API resource content to be fully represented in web search engine indexes [4].

To generate structured data, D109 would access the Registry to look for data sources, then retrieve from those sources either ARD or DRI, followed by structuring the data by making use of schemas found on schema.org and standardized Global Location Numbers (GLNs) from GS1. The search engines would eventually crawl the structured data and offer it to end users for them to find data more easily. The component and data flow is described in Figure 3, and the data generation process is described in [D109-process].

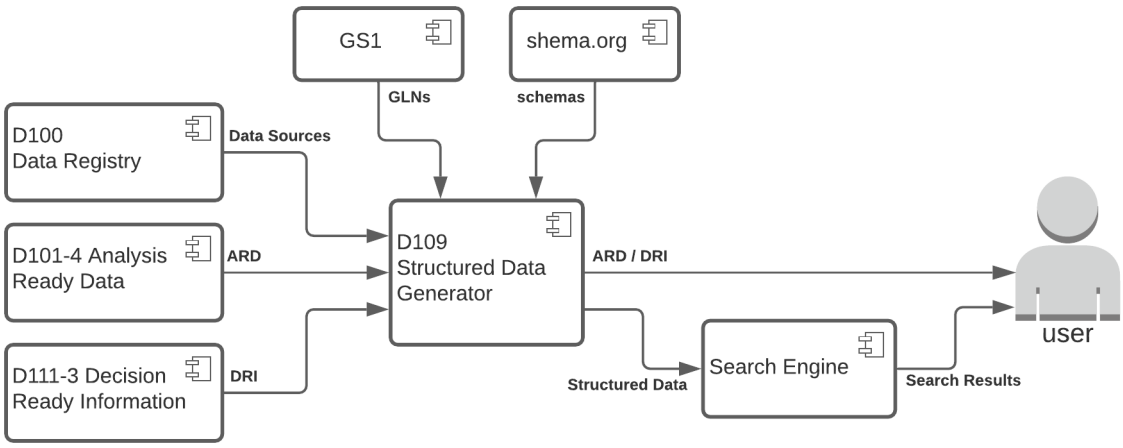

Figure 3 — D109 Data Flow

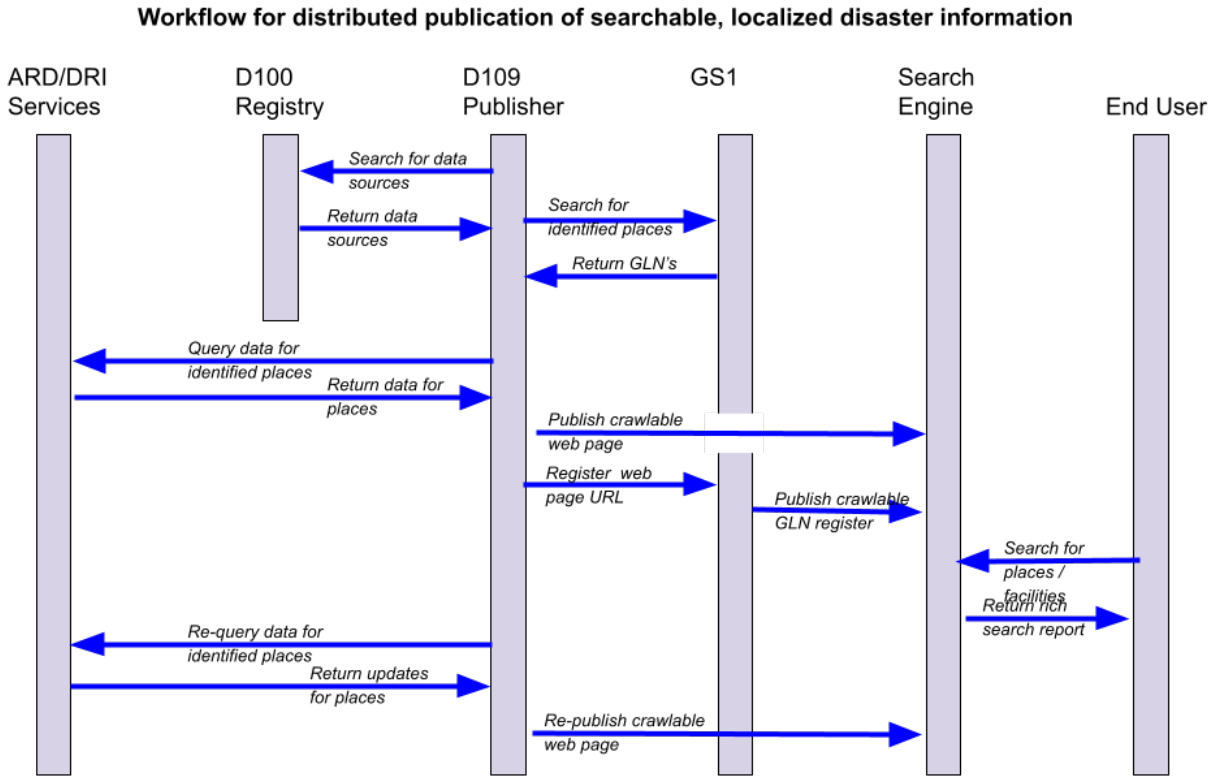

Figure 4 — D109 Process

7

STRUCTURED DATA GENERATOR

7 STRUCTURED DATA GENERATOR

This chapter describes the D109 component built for the Disaster Pilot by GeoSolutions. The goal of this component was to explore the potential for linked data to assist users in the use of search engines to search the web for information about disasters.

7.1. Status Quo

This section describes the current status regarding the use of structured data related to disaster management. The Disaster Pilot provided information on how GeoSolutions experimented using linked data in relation to disaster relevant datasets for the first time.

Previous work by GeoSolutions with linked data provided the basis for the work carried out for the pilot.

- The development of an extension for CKAN to expose and consume metadata from other catalogs using RDF documents serialized according to the Italian DCAT Application Profile. The Data Catalog Vocabulary (DCAT) is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. The extension was thus meant to promote the sharing and the standardization of metadata about the public dataset of the Italian public administration.
- The development of the GeoServer Features Templating plug-in, initially developed as a way to serve collections of Features as JSON-LD documents, which then was expanded to became a comprehensive tool to customize features output in a variety of formats (including GeoJSON, HTML, and GML).

7.2. Technical Architecture

To make it easier for users to search disaster information, it is critical that those publishing the data and those reading the data have a common understanding of the domain, which can be achieved through well-defined semantics that describe precisely how the published data should be interpreted without ambiguities.

GeoSolutions' approach to this task was to publish structured geospatial data using the JSON-LD format through a GeoServer OGC API — Features implementation. The JSON-LD context was defined based on Schema.org vocabulary.

The process consisted of the following steps.

- **Clause 7.2.1**: Find the proper web ontologies to map the source dataset to the chosen terms.
- **Clause 7.2.2**: Produce a JSON-LD output consistent with the datasets and the chosen vocabulary.
- **Clause 7.2.3**: Have the JSON-LD document in the <head> element of every HTML page.
- **Clause 7.2.4**: Validate the obtained JSON-LD document to ensure that the chosen terms were correctly mapped to the features' attributes.
- **Clause 7.2.5**: Submit the HTML to Google for page indexing.

Two datasets were used.

- **Mortality Risk Index (MRI) Dataset**: Dataset comprising metrics for mortality risk in the USA, supplied with statistical data about population (age, gender, household, ethnicity, and scholarization) as well as metrics about disease risk factors. Data were georeferenced using a polygonal theme representing Maryland administrative areas.
- **Transmission Risk Index (TRI) Dataset**: Dataset comprising metrics for COVID transmission risk in Perú in terms of number of cases, deaths, and active cases for various temporal periods. Data were georeferenced using a polygonal theme representing Perú administrative areas.

7.2.1. Choosing Vocabulary Types to Map Datasets

The aforementioned datasets represent well known metrics related to the TRI and MRI. Each record of both datasets is associated with a polygon representing the administrative boundary that the metric refers to. The first step was to map each one of those metrics with a well-known vocabulary in order to produce a structured output capable of holding the selected vocabulary semantics that could be then interpreted by search engines such as Google.

GeoSolutions used vocabularies from schema.org, which are supported by major search engines and web platforms, including Google. Ontologies like Simple Knowledge Organization System Reference (SKOS) and Dublin Core Metadata Initiative are well known and could have been used in parallel with schema.org to provide definitions of the same ontology with different terms. Nonetheless, no evidence was found related to the usage of these two Ontologies with JSON-LD by major search engines, since all the documentation found referred to the usage of schema.org. JSON-LD was the structured output format mainly due to it offering a lightweight JSON-based approach to encode linked data that allows users to choose the ontology that best fits their use case. JSON-LD is also the only format of structured data mentioned by Google[1] to improve SEO by including it in web pages.

For the MRI dataset no type was found that clearly referred to medical or population metrics while presenting at the same time a geographical property to refer the data to some geographical location. For this reason, the chosen type was Dataset. This type defines a generic

set of data through its property variableMeasured and at the same time includes a geographical property contentLocation to bind the dataset to a specific location in the form of a geometry.

For the TRI dataset the chosen type was SpecialAnnouncement. According to the type description provided in schema.org, the reference scenario of this type is the Coronavirus pandemic and is designed to communicate urgent information about COVID. As such, the type includes the property diseaseSpreadStatistics that provides a meaningful term to map the various metrics related to COVID transmission. Moreover, the type has the property spatialCoverage that allows the user to map the spatial reference present in the source data.

Take as an example the following JSON-LD excerpt:

```
{
"@context": {
    "schema": "http://schema.org/",
    "FeatureCollection": "http://schemas.opengis.net/wfs/2.0/wfs.xsd",
    "features":
    {
        "@container": "@set",
        "@id": "schema:hasPart"
    },
    "diseaseSpreadStatistics":
    {
        "@id": "schema:diseaseSpreadStatistics",
        "@container": "@index"
    },
    "containedInPlace": "schema:containedInPlace",
    "observedNode": "schema:observedNode",
    "populationType": "schema:populationType",
    "measuredValue": "schema:measuredValue",
    "measuredProperty": "schema:measuredProperty",
    "type": "schema:type",
    "value": "schema:value",
    "variableMeasured": "schema:variableMeasured",
    "identifier": "schema:identifier",
    "name": "schema:name",
    "description": "schema:description",
    "spatialCoverage": "schema:spatialCoverage",
    "area": "schema:additionalProperty",
    "geo": "schema:geo",
    "polygon": "schema:polygon",
    "datePosted": "schema:datePosted",
    "temporalCoverage": "schema:temporalCoverage"
},
"@type":"FeatureCollection",
"features":[
{
"@type":"schema:SpecialAnnouncement",
"identifier": "demo-peru-ppe-20210303.PE030101.2021-03-03",
"name": "COVID transmission risk ABANCAY",
"description": "COVID transmission risk",
"datePosted":"2021-11-24",
"diseaseSpreadStatistics":{
      "daily_cases":{
         "@type":"Observation",
         "observedNode":{
            "@type":"StatisticalPopulation",
            "@id":"#popType",
            "populationType":"Population of Lima"
         },
         "measuredProperty":"Daily Cases",
```

```
            "measuredValue":4000
         },
         "daily_deaths":{
            "@type":"Observation",
            "observedNode":{
               "@id":"#popType"
            },
            "measuredProperty":"Daily Deaths",
            "measuredValue":130
         }
      }
   }
 ]
}
```


Figure 5

From the JSON-LD excerpt shown above it is possible to see how a single geographical feature of the source dataset TRI looks once the dataset has been mapped to the target vocabulary. Each Feature has been assigned the type `SpecialAnnouncement`. This type has several properties like identifier, name, description, dateposted, and `diseaseSpreadStatistics`. Some of them are of simple type while others are defined using a complex type such as the `diseaseSpreadStatistics` property which is of type Observation. The type `Observation` has other properties like observedNode, measuredProperty and measuredValue that, in turn, can be of a complex type and have other nested properties like `observedNode`.

The term `FeatureCollection` mapped to the wfs.xsd schema IRI was used as the `@type` of the root JSON object of the JSON-LD output of both datasets. This was done due to an ontology definition problem: there was no ontology defined to have properties of type `SpecialAnnouncement` while at the same time having a geometric property for the administrative boundary polygon attribute. The ideal situation would have been to be able to use the `SpecialAnnouncement` as the `@type` of the root JSON object while mapping the features array attribute to the `diseaseSpreadStatistics` property, having each Feature considered of `@type` Observation. However, the fact that a geometry property is available only for the `SpecialAnnouncement` type and not for the Observation type, GeoSolutions was forced to consider every single Feature as a `SpecialAnnouncement`, impeding the use of it for the root JSON object. Furthermore, no other schema.org type was found having properties of type `SpecialAnnouncement`, forcing the use of an unrelated root `@type` (chosen to be FeatureCollection) and the generic property hasPart for the features attribute names. The same limitation was encountered for the use of the `Dataset` type. This issue will be further detailed in the next sections which describe the proper map terms and properties for the JSON-LD context.

Table 1 summarizes the TRI dataset attributes and corresponding vocabulary terms mapped from schema.org. Please note that `SpecialAnnouncement` is the root type, and that the 70 metrics have been summarized into one row within the table as these metrics were mapped to the same vocabulary type.

Table 1 — TRI Dataset Attributes and its Mapped Vocabulary Types

ATTRIBUTE	METRIC DESCRIPTION	METRIC CATEGORY	VOCABULARY TYPES
Geom_Level	Geometry Level	Identification Field	property: `spatialCoverage` (type AdministrativeArea) → `description`

ATTRIBUTE	METRIC DESCRIPTION	METRIC CATEGORY	VOCABULARY TYPES
Geom_ID	Geometry ID	Identification Field	property: `identifier`
Geom_Name	Geometry Name	Identification Field	property: `spatialCoverage` (type `AdministrativeArea`) → `name`
Geom	Geometry of the administrative area	Geometry field	property: `spatialCoverage` (type `AdministrativeArea`) → geo (type GeoShape) → polygon
ADM2_ES	Administrative unit name	Administrative unit name	property: `spatialCoverage` (type `AdministrativeArea`) → containedInPlace (type Place) → `name`
ADM1_ES	Administrative unit name	Administrative unit name	SAME AS ABOVE
REC_Date	Date	Identification Field	NOT USED
46 different metrics	46 different metrics	Cumulative Metric, Daily Metric, Rolling Average Metric, Crude Fatality Rate Metric, Change Rate Metric, Population, Geographic Information,	Transmission Risk Index

Table 2 summarizes the MRI dataset attributes and corresponding vocabulary terms mapped from schema.org. Please note that `Dataset` is the root type, and that the 70 metrics have been summarized into one row within the table as these metrics were mapped to the same vocabulary type.

Table 2 — MRI Dataset Attributes and its Mapped Vocabulary Types

ATTRIBUTE	METRIC DESCRIPTION	METRIC CATEGORY	VOCABULARY TYPES
Geom_Level	Geometry Level	Identification Field	property: `contentLocation` (type `AdministritativeArea`) → `description`
Geom_ID	Geometry ID	Identification Field	property: `identifier`
Geom_Name	Geometry Name	Identification Field	property: `contentLocation` (type `AdministritativeArea`) → `name`
ALAND_MI	Total Land Area in Miles	Geographic Information	NOT USED
AWATER_MI	Total Water Area in Miles	Geographic Information	NOT USED

ATTRIBUTE	METRIC DESCRIPTION	METRIC CATEGORY	VOCABULARY TYPES
ATOTAL_MI	Total Area in Miles	Geographic Information	property: `contentLocation` (type `AdministritativeArea`) → additional Property (type PropertyValue) → value
Geom	Geometry of the administrative area	Geometry field	property: `contentLocation` (type `AdministrativeArea`) → `geo` (type `GeoShape`) → `polygon`
ADM2_ES	Administrative unit name	Administrative unit name	property: `contentLocation` (type `AdministrativeArea`) → `containedInPlace` (type `Place`) → `name`
ADM1_ES	Administrative unit name	Administrative unit name	SAME AS ABOVE
ADM0_ES	Administrative unit name	Administrative unit name	SAME AS ABOVE
70 different metrics	70 different metrics	SDOH Total, Sex, Age, Race, Housing Tenancy, Housing Density, Income, Education, Fluency, SNAP, Employment, Occupation, Underlying Health Condition, Mortality Risk Index	property: `variableMeasured` (type → `PropertyValue`) → `value`

The process of mapping vocabularies to the datasets revealed challenges, which are detailed in Finding Appropriate Vocabularies. As a result, the need for finding vocabularies that better match the semantics still remains.

7.2.2. Generating the JSON-LD Output

JSON-LD documents require a context definition which can be inline or referenced. The context defines the vocabulary used to encode the data semantics. Each attribute name is mapped to an Internationalized Resource Identifier or IRI, in this case the schema.org URL, pointing to its semantic definition. As an example, the following code block is the context used for the TRI dataset: zdfg

```
"@context": {
    "schema": "http://schema.org/",
    "FeatureCollection": "http://schemas.opengis.net/wfs/2.0/wfs.xsd",
    "features":
    {
        "@container": "@set",
        "@id": "schema:hasPart"
    },
    "diseaseSpreadStatistics":
    {
        "@id": "schema:diseaseSpreadStatistics",
        "@container": "@index"
    },
    "containedInPlace": "schema:containedInPlace",
```

```
    "observedNode": "schema:observedNode",
    "populationType": "schema:populationType",
    "measuredValue": "schema:measuredValue",
    "measuredProperty": "schema:measuredProperty",
    "type": "schema:type",
    "value": "schema:value",
    "variableMeasured": "schema:variableMeasured",
    "identifier": "schema:identifier",
    "name": "schema:name",
    "description": "schema:description",
    "spatialCoverage": "schema:spatialCoverage",
    "area": "schema:additionalProperty",
    "geo": "schema:geo",
    "polygon": "schema:polygon",
    "datePosted": "schema:datePosted",
    "temporalCoverage": "schema:temporalCoverage"
}
```

Figure 6

A JSON-LD context must map each property and type used in the JSON-LD document to a valid IRI through the use of a term. A type can be thought of as an ontology definition, e.g., `SpecialAnnouncement` is a type, to which a JSON object refers to via the `@type` attribute. A property is a property of a specific ontology which in turn has its own type, e.g., `diseaseSpreadStatistics` is a property of `SpecialAnnouncement` and can assume the type `Observation` among the others.

7.2.2.1. Preserving Querying Capabilities Through Index Maps

Preserving the querying capability offered by features templating emerged as a functional problem when defining a JSON-LD template to match the datasets with the chosen ontology. Templating those metrics as a JSON array with several JSON objects of type `Observation` would have eliminated the possibility of querying the data due to missing unique attribute names for each metric encoded as an `Observation`.

Each metric was ultimately assigned to the property `diseaseSpreadStatistics` of the type `Observation`. To do this, **index maps** were used. An index map allows keys that have no semantic meaning (i.e., that cannot be mapped to any term), but should be preserved in the output, to be used in JSON-LD documents. This is useful when the data being used in the template has multiple source attributes that we want to map to the same property semantic. In this case, many properties were considered semantically equivalent to the `diseaseSpreadStatistics` property of the `SpecialAnnouncement` type. Therefore, GeoSolutions used an index map in the `@context`, where `@id` references the `diseaseSpreadStatistics` and the type of the container is set to `@index`:

```
"diseaseSpreadStatistics":{
      "@id":"http://schema.org/diseaseSpreadStatistics",
      "@container":"@index"
   },
```

Figure 7

This definition allows the use of unbounded JSON attributes names in the `@context` if they are defined inside a `diseaseSpreadStatistics` object. For example:

```
"diseaseSpreadStatistics":{
      "daily_cases":{
         "@type":"Observation",
         "observedNode":{
            "@type":"StatisticalPopulation",
            "@id":"#popType",
            "populationType":"Population of Lima"
         },
         "measuredProperty":"Daily Cases",
         "measuredValue":4000
      },
      "daily_deaths":{
         "@type":"Observation",
         "observedNode":{
            "@id":"#popType"
         },
         "measuredProperty":"Daily Deaths",
         "measuredValue":130
      },
    },
```

Figure 8

In this case, attribute names `daily_cases` and `daily_deaths` were not associated with any term in the `@context`, yet the JSON-LD was valid. This choice also had a functional aspect: by being able to preserve original attribute names through the use of an index map, it was not needed to define the object of type `diseaseSpreadStatistics` as a JSON Array. By doing this, it was possible to take full advantage of features templating backwards mapping.

7.2.2.2. Increasing JSON-LD Generation Efficiency Through Features Templates

After the vocabulary has been chosen and the context has been defined, the features need to be served as a JSON-LD document matching the chosen terms.

Features are normally served by GeoServer following the schema found in the datasource itself. In case of a database source, the schema is normally constituted by the definition of the database's tables. The GeoJSON encoder would then parse the feature to encode the geometry and the ID as top level attributes of a single GeoJSON feature and the rest of the value as attributes of the JSON Object `properties`.

When dealing with JSON-LD, attribute names and their level of nesting inside the final output are not given in advance by any specification as they depend upon the chosen vocabulary. This causes a JSON-LD document structure to require defining several nested JSON objects and arrays according to the chosen vocabulary. This process must be repeated for every feature, i.e., each metric, which can be time-consuming. To speed this process, GeoSolutions used the GeoServer Features Templating Plugin. The plugin allows users to define a JSON-LD template document where feature properties references can be placed in any JSON structure of choice allowing a fine grained control over the final output through a What You See Is What You Get (WYSWYG) approach (see Figure 9).

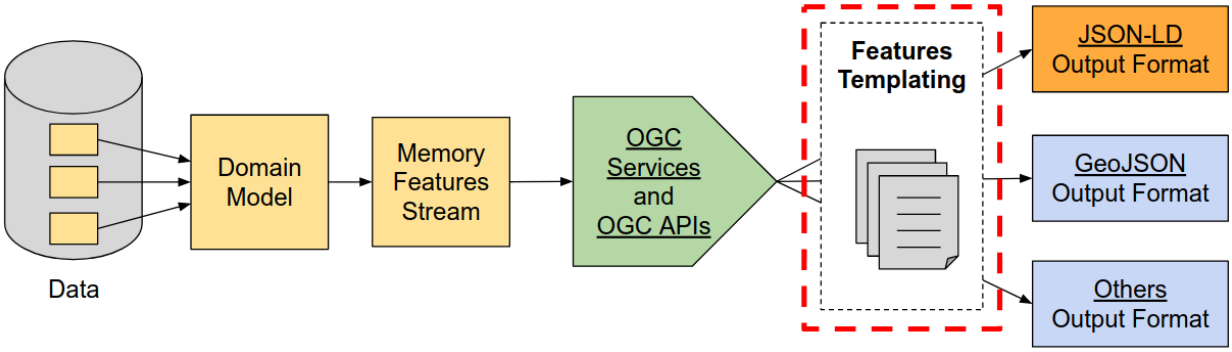

Figure 9 — GeoServer Features Templating Plugin Architecture Overview

Features Templating provides a template for a single feature and then applies the template to all the features present in the streams of data being encoded. The templates can be created directly using the GeoServer web user interface.

The Features Templating GeoServer UI allows the user to speed up the process of creating a JSON-LD document by making several operations easy while configuring a template (see Figure 10).

- It provides a Text Editor to write the template.
- It allows template developing while testing with the preview functionality.
- It provides validation over the output produced by the template.

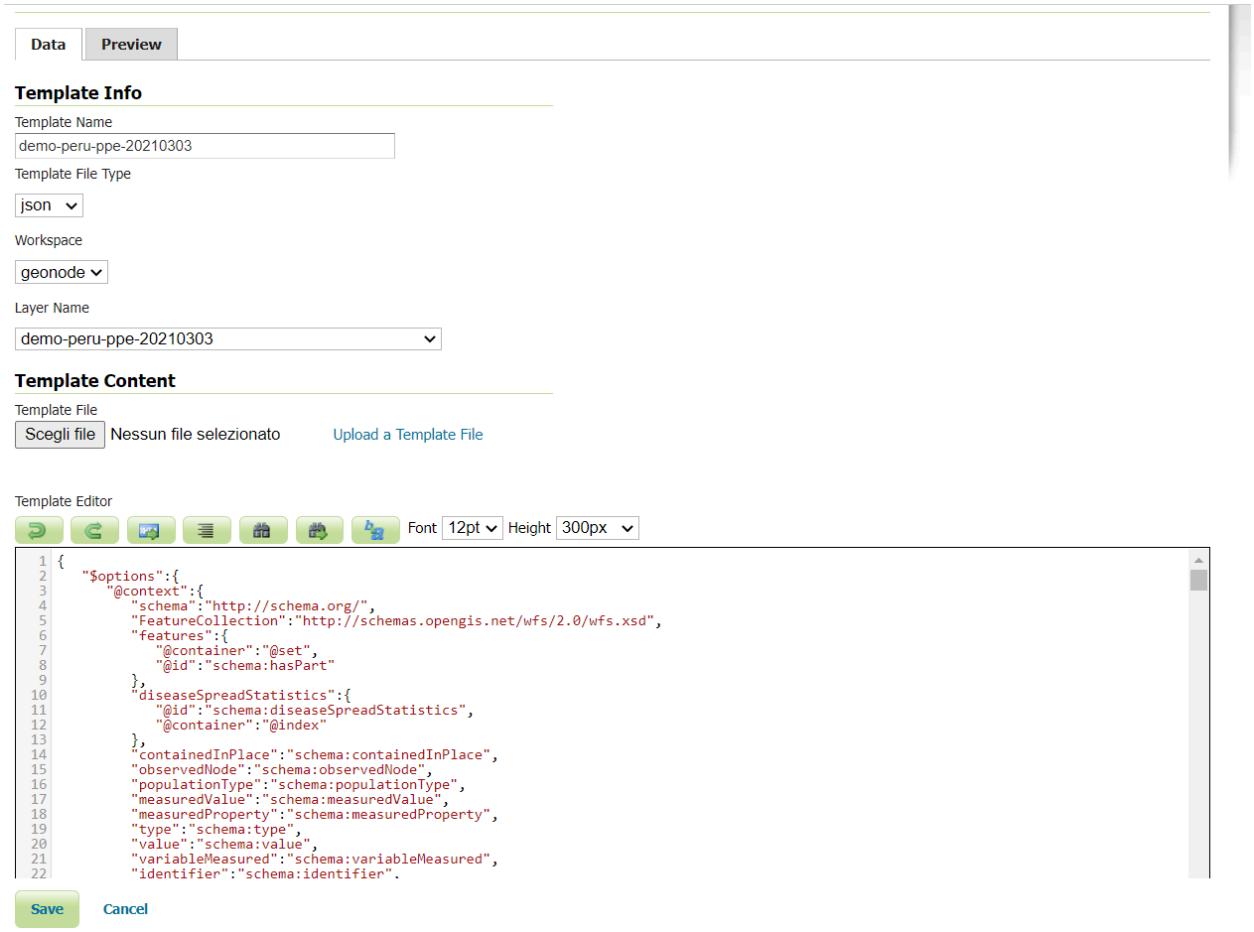

Figure 10 — GeoServer Features Templating Configuration Page

Templates are made up of directives, which dictate the behavior of the template. Directives are used to interpolate, transform, and filter values coming from the data being templated.

Features Templating also provides more advanced directives that are outside the scope of this document. Any other part of the template that is not a directive will be encoded as it is in the final output.

Directives can be of two types.

- **Vendor options**: Directives that allow output customization outside the scope of a feature. As an example, the JSON-LD `@context` is provided through an option since it doesn't affect the encoding of the features, but instead provides a JSON object to be inserted at the beginning of the output.

- **Property directives**: Directives that allow the manipulation of Feature properties encoding by invoking them in the desired point of the template through the property interpolation directive (e.g., `${propertyName}`) and by further processing the property's value through CQL functions using the CQL directive (e.g., `$${strConcat('custome_prefix', propertyName)}`).

The following code block is the template created for the TRI dataset explaining the rationale behind it. Given the number of properties present in the example dataset, some repeating parts have been removed from the original template for the sake of clarity. As most metrics share the definitions of `@type` and `observedNode`, only a few were left in this example.

```
{
   "$options":{
      "@context":{
         "schema":"http://schema.org/",
         "FeatureCollection":"http://schemas.opengis.net/wfs/2.0/wfs.xsd",
         "features":{
            "@container":"@set",
            "@id":"schema:hasPart"
         },
         "diseaseSpreadStatistics":{
            "@id":"schema:diseaseSpreadStatistics",
            "@container":"@index"
         },
         "containedInPlace":"schema:containedInPlace",
         "observedNode":"schema:observedNode",
         "populationType":"schema:populationType",
         "measuredValue":"schema:measuredValue",
         "measuredProperty":"schema:measuredProperty",
         "type":"schema:type",
         "value":"schema:value",
         "variableMeasured":"schema:variableMeasured",
         "identifier":"schema:identifier",
         "name":"schema:name",
         "description":"schema:description",
         "spatialCoverage":"schema:spatialCoverage",
         "area":"schema:additionalProperty",
         "geo":"schema:geo",
         "polygon":"schema:polygon",
         "datePosted":"schema:datePosted",
         "temporalCoverage":"schema:temporalCoverage"
      },
      "@type":"FeatureCollection"
   },
   "@type":"schema:SpecialAnnouncement",
   "identifier":"${@id}",
   "name": "$${strConcat('COVID transmission risk ',Geom_Name)}",
   "description": "COVID transmission risk",
   "datePosted":"2021-11-24",
   "diseaseSpreadStatistics":{
      "@type":"schema:Observation",
      "daily_cases":{
         "@type":"schema:Observation",
         "observedNode":{
            "@type":"schema:StatisticalPopulation",
            "@id":"#popType",
            "populationType":"$${strConcat('Population of ', Geom_Name)}"
         },
         "measuredProperty":"Daily Cases",
         "measuredValue":"${Daily_Cases}"
      },
      "daily_deaths":{
         "@type":"schema:Observation",
         "observedNode":{
            "@id":"#popType"
         },
```

```
            "measuredProperty":"Daily Deaths",
            "measuredValue":"${Daily_Deaths}"
         },
         "daily_active":{
            "@type":"schema:Observation",
            "observedNode":{
               "@id":"#popType"
            },
            "measuredProperty":"Daily Active Cases",
            "measuredValue":"${Daily_Active}"
         },
         "cum_cases":{
            "@type":"schema:Observation",
            "observedNode":{
               "@id":"#popType"
            },
            "measuredProperty":"Cumulative cases",
            "measuredValue":"${Cum_Cases}"
         },
         [...]
      },
         "spatialCoverage":{
            "@type":"schema:AdministrativeArea",
            "name":"${Geom_Name}",
            "description":"${Geom_Level}",
            "geo":{
               "@type":"schema:GeoShape",
               "polygon":"$${toWKT(geom)}"
            },
            "containedInPlace":{
               "@type":"schema:Place",
               "name":"${ADM2_ES}",
               "containedInPlace":{
                  "name":"${ADM1_ES}",
                  "containedInPlace":{
                     "name":"${ADM0_ES}"
                  }
               },
               [...]
            },
            "area":[
               {
                  "@type":"schema:PropertyValue",
                  "name":"Total Land Area in Miles",
                  "value":"${KM2}"
               }
            ]
      }
   }
```

Figure 11 — TRI Dataset Template

The first element of the template, the `$option` attribute, is a directive and is defined as a JSON object that contains all the vendor options that will be used to customize the part of the output outside the feature scope. In this case, the vendor options included the JSON object `@context` and the attribute of the root JSON object `@type`. The `@type` attribute value should always be mapped to an IRI found in the reference vocabulary. Furthermore, the `@context` mapped the term `FeatureCollection` to the `wfs.xsd` schema (`FeatureCollection":"http://schemas.opengis.net/wfs/2.0/wfs.xsd"`).

The subsequent parts of the template customize the appearance of each feature in the resulting features array. The `@type` attribute was defined for each Feature to be of type `SpecialAnnouncement` (the `SpecialAnnouncement` term is correctly mapped to an IRI).

Each term used as a property name for each Feature must be correctly referenced in the `@context`. An exception to this rule is when index maps are used (see above) as index maps do not require mapping the attribute name to an IRI.

In the case of the type `SpecialAnnouncement`, based on the description provided for each of its properties, the values were then mapped to them in the template using the various directives like in the following examples.

- The name property is simply described as the name of the item. Each Feature as a type `SpecialAnnouncement` has then been assigned with the name constructed by the String 'COVID transmission risk' and the value Geom_Name coming from the source data that refers to the place name.
- The identifier property of `SpecialAnnouncement` was used to refer to Feature @id.
- The description was assigned with a static value describing the Feature as related to Covid transmission risk data.
- The `diseaseSpreadStatistics` property has been used to reference the various COVID related metrics in the dataset. Since the object `diseaseSpreadStatistics` was defined as an index @container in the `@context`, it was not necessary to map the attributes name to an IRI inside the `diseaseSpreadStatistics` object.
    - As a property of type `Observation`, a `diseaseSpreadStatistics` instance can have, in turn, other properties.
        - `observedNode`: Described by schema.org as the Statistical population and thus obtained by a String concatenation of 'Population of' and the Geom_Name attribute.
        - `measuredProperty`: Assigned with a static string defining what the measurement is about.
        - `measuredValue`: The value of the observation taken if the template has been obtained by using a property interpolation directive over the various metrics in the source dataset.

- The property `spatialCoverage` indicates the place that is the focus of the content of an object of type `SpecialAnnouncement`.
    - As a property of type `AdministrativeArea`, `spatialCoverage`, it has, in turn, several properties.
        - The `geo` property can contain the geometry theme of the place as a `polygon`. To map it, a CQL directive was used to transform the geometry to a WKT representation of it through the expression (`$${toWKT(geom)}`).
        - The `containedInPlace` property is inherited by the supertype `Place` and expresses a recursive relation inside it since each `containedInPlace` is of type `Place` and can have in turn another `containedInPlace` property. It has been used to map the source data regarding the administrative units expressed hierarchically through a containment relation. For units hierarchically superior to the one represented by the geometry value, only the property `name` has been defined since there was no geographic data related to it.
    - `AdministrativeArea` also has a generic property `additionalProperty` described as a property-value pair representing an additional characteristic of the entity for which there is no matching property in schema.org. Due the lack of terms in `AdministrativeArea` for a property related to the metric extension of the geometry, the decision was made to map the term `area` to the IRI of `additionalProperty` and use it in the template to provide the value of the Land Area in Miles of the related geometry.
        - `additionalProperty` is a generic property that should only be used when strictly needed since applications designed to use specific schema.org properties will typically expect such data to be provided using those properties, rather than using the generic property/value mechanism. These means that consumers of the JSON-LD might end up not using it

The same rationale was applied to the Mortality Risk dataset. The metrics were all assigned to the same property/type (`variableMeasured` with type `PropertyValue`) and an expression `@container:index` was again defined in the `@context` to leverage the index map and preserve query capabilities on the layer. The property `contentLocation` of type `Place`, which is inside the `spatialCoverage` object, was used for the geospatial data.

```
Mortality risk index dataset GeoServer JSON-LD template:
{
   "$options":{
      "@context":{
         "features":{
            "@container":"@set",
            "@id":"schema:hasPart"
         },
         "Feature":"http://schemas.opengis.net/gml/2.1.2/feature.xsd",
         "schema":"http://schema.org/",
         "containedInPlace":"schema:conatainedInPlace",
         "value":"schema:value",
```

```
      "name":"schema:name",
      "about":"schema:about",
      "type":"schema:type",
      "value":"schema:value",
      "variableMeasured": {
         "@id":"schema:variableMeasured",
         "@container":"@index"
      },
      "identifier":"schema:identifier",
      "name":"schema:name",
      "description":"schema:description",
      "spatialCoverage":"schema:spatialCoverage",
      "area":"schema:additionalProperty",
      "geo":"schema:geo",
      "polygon":"schema:polygon",
      "temporalCoverage":"schema:temporalCoverage"
   }
},
"@type":[
   "Feature",
   "Dataset"
],
"identifier":"${@id}",
"about":{
   "@type":"Thing",
   "name":"COVID mortality risk"
},
"variableMeasured":{
   "ep_pop":{
      "@type":"PropertyValue",
      "name":"Total population",
      "value":"${E_POP}"
   },
   "ep_male":{
      "@type":"PropertyValue",
      "name":"Percent Male",
      "value":"${EP_Male}"
   }
   [...]
},
"contentLocation":{
      "@type":"Place",
      "name":"${Geom_Name}",
      "description":"${Geom_Level}",
      "geo":{
         "@type":"GeoShape",
         "polygon":"$${toWKT(geom)}"
      },
      "containedInPlace":{
         "@type":"Place",
         "name":"${ADM2_ES}",
         "containedInPlace":{
            "name":"${ADM1_ES}",
            "containedInPlace":{
               "name":"${ADM0_ES}"
            }
         }
      },
      "area":{
         "@type":"PropertyValue",
         "name":"Total Land Area",
         "value":"${Shape_Area}"
      }
```

```
        }
    }
}
```

Figure 12 — MRI Dataset Template

Once the template is correctly configured it must be associated with a particular **content negotiation rule** which triggers a template on a particular layer based on some condition. The simplest condition is the output format matching, so that a JSON-LD template is applied to a layer only when the requested output format is `ld+json` while a GeoJSON template will be applied when the output format is `geo+json`. More complex rules are possible.

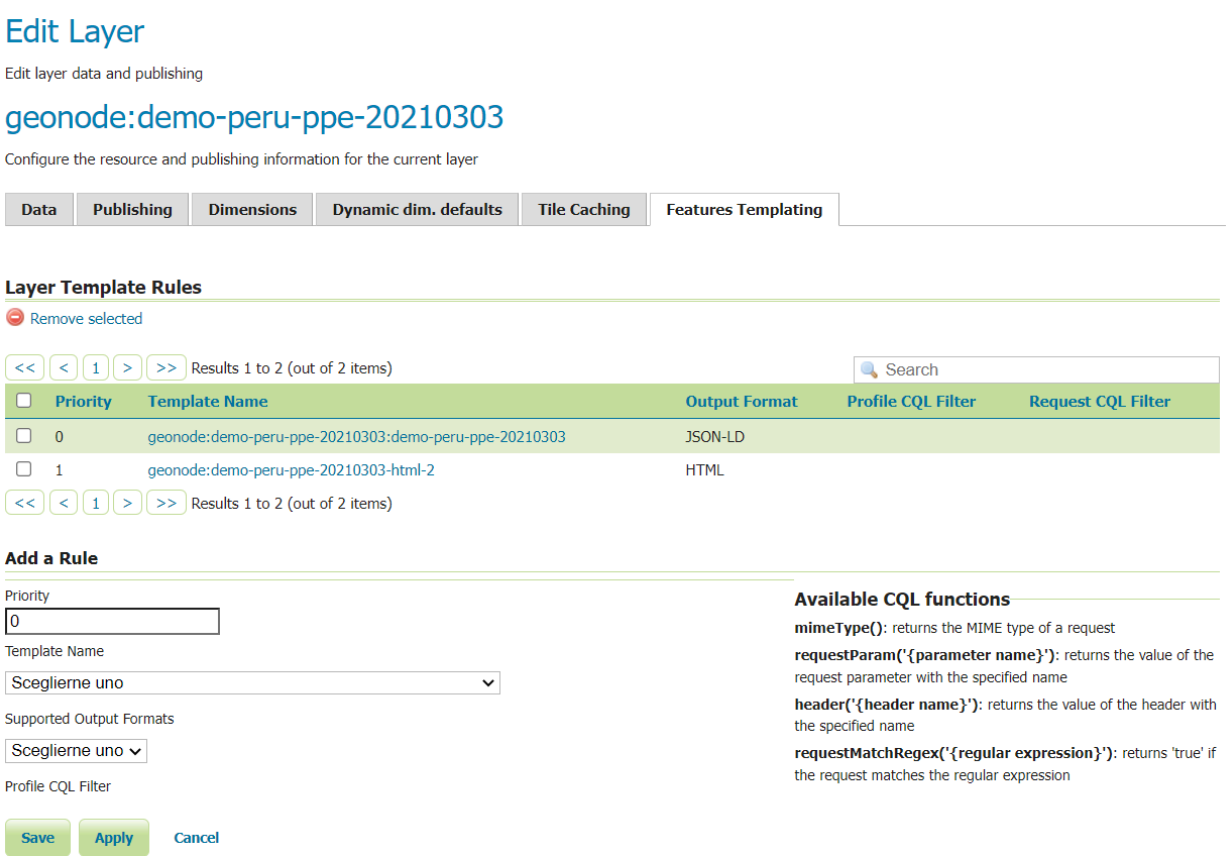

Figure 13 — Template Rules Configuration Page

As seen on Figure 13, GeoSolutions configured a rule that triggers the JSON-LD template named demo-peru-ppe-20210303 whenever the requested output format was `ld+json`. Once it was configured, it was possible to successfully retrieve the `demo-peru-ppe-20210303` (TRI dataset) layer as a JSON-LD.

7.2.3. Injecting the JSON-LD into an HTML Document

Once the JSON-LD content has been generated, it needs to be injected into an HTML output in order to allow Google to consume it[1]. A JSON-LD document needs to be present inside the `<head>` element of an HTML page inside an element `<script type="application/ld+json">`.

GeoSolutions explored a GeoServer functionality to inject JSON-LD documents into HTML outputs through the creation of HTML templates that output features using a tree-like structure. The GeoServer documentation specifies how to create these templates for them to work together with JSON-LD templates in order to return features in JSON-LD.

As seen on Figure 14, the tag `<script type="application/ld+json"/>` is added to allow injecting the JSON-LD representation of the features being templated in the <head>. Each metric is included in this template through a combination of tags `<ul>` and `<li>`.

Figure 14 — The HTML Template Configured to Have the JSON-LD Output Injected

Figure 15 — The HTML Document With the ld+json Script Element

The HTML output can be seen in Figure 15. The HTML code block shown below gives a closer look at the script element holding the JSON-LD (the JSON-LD output and the HTML body have been truncated for clarity).

```
<html>
  <head>
    <script type="application/ld+json">
      {"@context":{"schema":"http://schema.org/","FeatureCollection":
"http://schemas.opengis.net/wfs/2.0/wfs.xsd","features":{"@container":
```

```
"@set","@id":"schema:hasPart"},"diseaseSpreadStatistics":{"@id":"schema:
diseaseSpreadStatistics","@container":"@index"},"containedInPlace":"schema:
containedInPlace","observedNode":"schema:observedNode","populationType":
"schema:populationType","measuredValue":"schema:measuredValue",
"measuredProperty":"schema:measuredProperty","type":"schema:type","value":
"schema:value","variableMeasured":"schema:variableMeasured","identifier":
"schema:identifier","name":"schema:name","description":"schema:description",
"spatialCoverage":"schema:spatialCoverage","area":"schema:additionalProperty",
"geo":"schema:geo","polygon":"schema:polygon","datePosted":"schema:datePosted",
"temporalCoverage":"schema:temporalCoverage"},"type":"FeatureCollection",
"@type":"FeatureCollection","features":[{"@type":"schema:SpecialAnnouncement",
"identifier":"demo-peru-ppe-20210303.PE030101.2021-03-03","name":"COVID
trasmission risk ABANCAY","description":"COVID trasmission risk","datePosted":
"2021-11-24","diseaseSpreadStatistics":{"@type":"schema:Observation",
"daily_cases":{"@type":"schema:Observation","observedNode":{"@type":"schema:
StatisticalPopulation","@id":"#popType","populationType":"Population of
ABANCAY"},"measuredProperty":"Daily Cases","measuredValue":10.0},"daily_
deaths":{"@type":"schema:Observation","observedNode":{"@id":"#popType"},
"measuredProperty":"Daily Deaths","measuredValue":0.0} [...]
    </script>
  </head>
  <body>[...]</body>
</html>
```


Figure 16

7.2.4. Validating the HTML with the JSON-LD

Validating the webpage and its JSON-LD context can help identify problems with the JSON-LD before publishing the HTML. It consists of verifying that the JSON-LD document is respecting the specification and that every JSON attribute is correctly referenced in the context and thus mapped to an IRI. Two validation tools were examined: Clause 7.2.4.1 and Clause 7.2.4.2.

7.2.4.1. Google Validator

Google provides a validation tool that identifies which rich result types were found on the page, as well as any errors or suggestions for its structured data. The Google validator checks the `@type` definition in the JSON-LD document against the one on the schema.org website. The validator raises an error in case a mandatory property of a type is missing, and raises a warning in case an optional property of a type is not defined. The tool is thus useful in understanding if the property and the type are correctly being mapped to an IRI and if any of the mandatory property is missing. The tool distinguishes between the properties that the schema.org `@type` mandates and the ones that are optional.

During the Pilot, the HTML output generated in the previous step was validated with the Google tool, as seen on Figure 17. The HTML page generated for this Pilot was found to be valid with respect to its JSON-LD embedded output. The validation also returned two warnings for two fields that had not been added to the JSON-LD template despite being present in the `SpecialAnnouncement` type on schema.org.

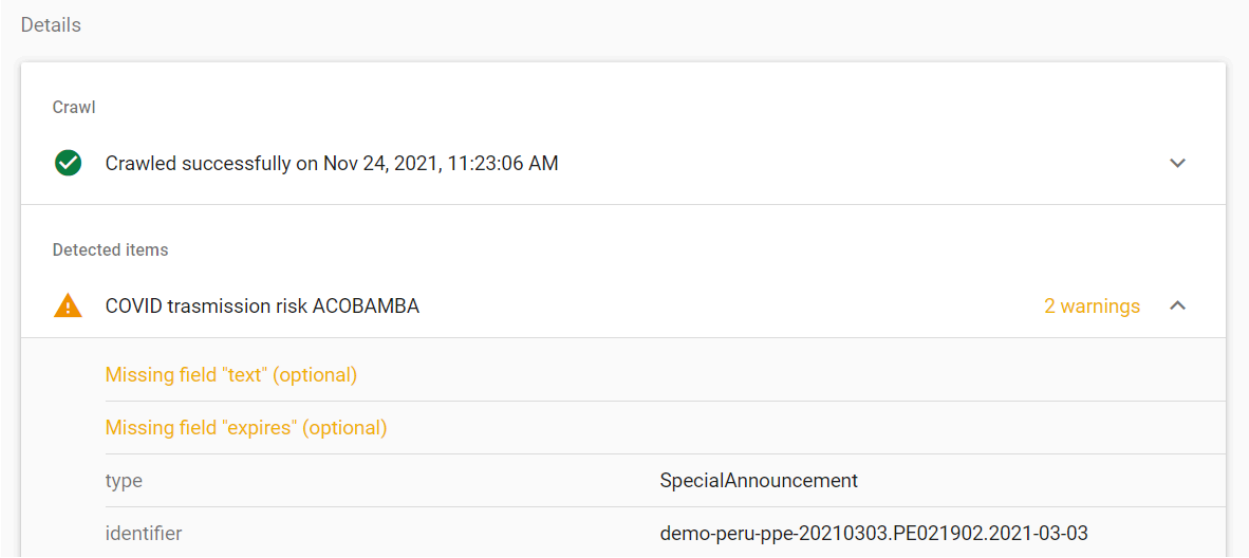

Figure 17 — Google JSON-LD Validation

The Validator enables users to preview the search result, which can be seen on Figure 18. This preview is useful in checking if the JSON-LD is used by Google to enable special search results. In this case, probably due to the simplicity of the HTML page used, no special search results were displayed.

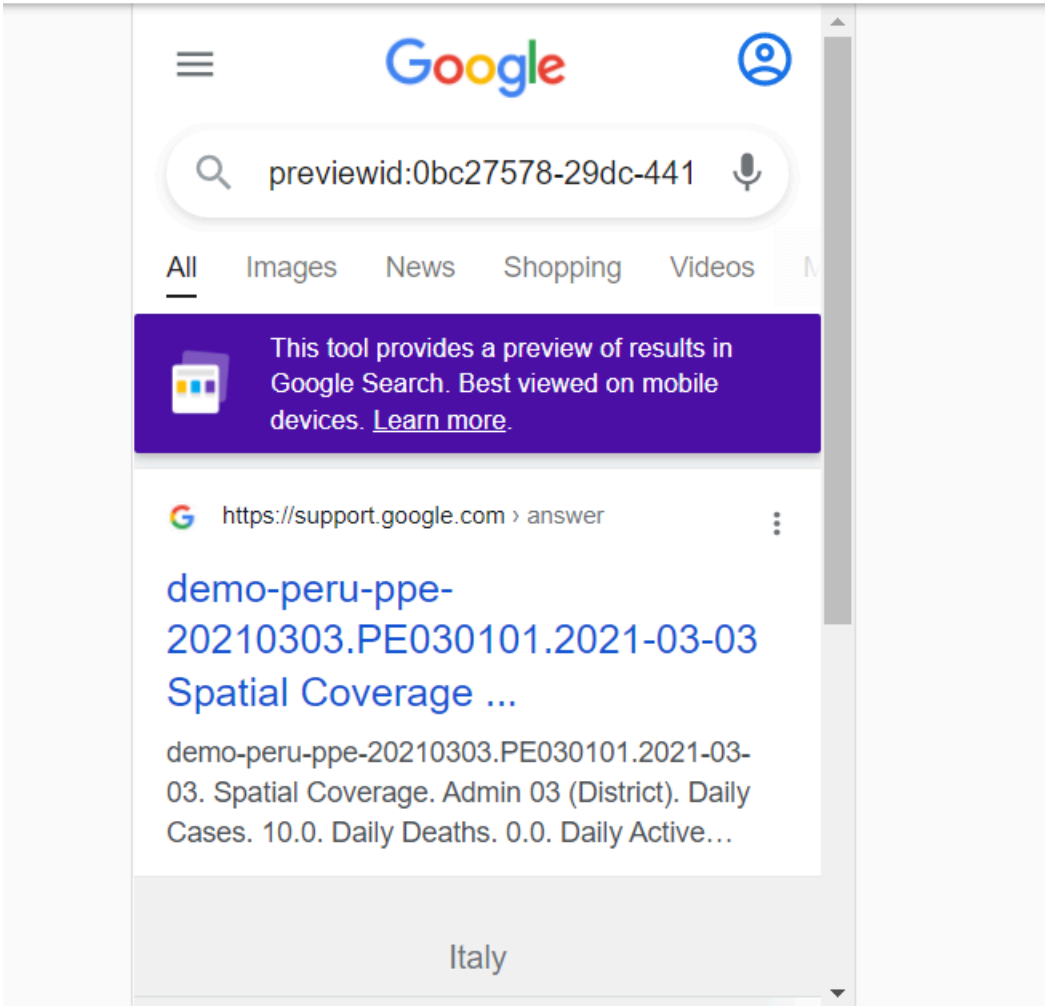

Figure 18 — Search Preview

7.2.4.2. GeoServer Validator

Another validation method used was the one included in the GeoServer Features Templating Plugin. The plugin includes a button on the preview tab which triggers a JSON-LD specific validation that checks that all the properties are correctly referenced in the `@context`. The validation works by using two JSON-LD algorithms named expansion and compaction. The expansion algorithm is run first and works by removing the `@context` from the JSON-LD and replacing any property in the JSON-LD with the mapped IRI. The compaction algorithm does the reverse and works by recreating the `@context` from an expanded JSON-LD document. After the two algorithms are executed, the result is compared with the original JSON-LD, and if any property present in the original JSON-LD is missing, it means that some term was not correctly mapped to an IRI and has been erased during the JSON-LD expansion. The preview tool is then able to inform the user of the name of the properties not correctly referenced by an IRI.

Figure 19 — GeoServer JSON-LD Validation

As seen on Figure 18, the GeoServer validation is limited to verifying that each `@type` and property in the JSON-LD document is correctly mapped to an IRI in the `@context`. As such, no validation is done against the type definition available at the matching IRI as the Google validation does. Therefore, the GeoServer validation is mainly meant to have a quick and first assessment on the validity of the JSON-LD document in terms of syntax and correctness of the `@context`. For a complete semantic validation, a more comprehensive validation tool such as the Google validator is required.

7.2.5. Submitting the page for Google Indexing

The final step is to publish the web page to make it accessible for the crawlers of search engines. Crawlers automatically go through HTML files and extract from the embedded JSON-LD the data required to feed the indexes that the search engines use to return search results.

In order to submit the page to Google for indexing it was necessary to:

1. go to the Google Search Console, copy and paste the URL pointing to the static HTML page; and
2. choose a verification method among the ones available and verify the submitted HTML.

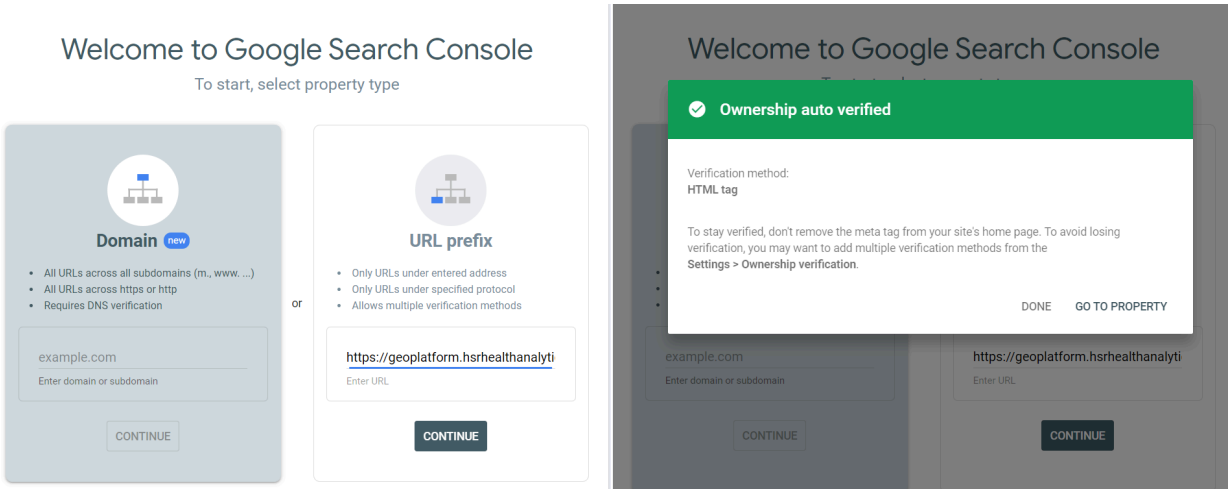

Figure 20 — Google Search Console

8

EXPLORATIONS ON GLOBAL LOCATION NUMBERS

8 EXPLORATIONS ON GLOBAL LOCATION NUMBERS

Both GS1 US and GS1 Global Office offered contributions in kind to the 2021 OGC Disasters Pilot. GS1 US has been working with OGC for a couple of years discussing the value of partnering some of our standards. For example, OGC has standards that require the identity of a thing or a location where it would be valuable to use an already recognized ubiquitous identifier such as GS1 identifiers.

Specific to the 2021 OGC Disasters Pilot, GS1 discussed the use of the GS1 Global Location Number (GLN) for facility and location identification as well as a paper prototype and discussion of the use of GS1 Digital Link to access available disaster related information by location.

GS1 US worked with Health Solutions Research successfully finding 29 of 35 hospitals in New Orleans and Maryland in the United States that already have a GS1 Global Location Number, demonstrating the value of an identifier already used by many in the healthcare space. Using a globally unique and recognized location identifier such as GS1 GLN to identify facilities and locations can help to more easily enable access to related information about that facility or hospital.

Incorporating GLN into GS1 Digital Link standard URI syntax as the primary key allows easy linked access to existing information on the internet that these facilities provide specific to what is needed in the event of a disaster such as number of available beds, number of available ventilators, PPE equipment available, location in relation to the disaster itself, etc. This has the greatest potential if that information is included as structured data embedded in Web pages. The GS1 Links registry has an associated sitemap that provides an index of all the available sets of links, which are expressed in JSON-LD, so that it acts as a node in the Open Data Cloud based on GS1 identifiers. This is undergoing further testing by GS1 Global Office to prove what they believe will be a benefit for searching regarding web services.

In addition, the ability of Skymantics to utilize GLN and related information to route the best path for suppliers to access a hospital or other facility during the event of a disaster was discussed. GLN standardized attribution and data quality rules can also provide benefit to this type of scenario in addition to accessing expanded disaster related information.

9

LESSONS LEARNED / CHALLENGES

9 LESSONS LEARNED / CHALLENGES

This chapter describes challenges and lessons learned that emerged from the efforts carried out during the pilot.

9.1. Serving Geographic data With JSON-LD

GeoServer OGCAPI — Features proved to be very powerful in providing access to geographic data, since it leverages all the capabilities that GeoServer provides with standard OGC services in the context of resource-centric APIs that take advantage of modern web development practices. As such, it allows an easy switch among different media types with proper content negotiation definition. For this reason, the JSON-LD output format fits well in its context.

The GeoServer Features templating plugin proved to be helpful at allowing users to produce templates on a per format basis. It made it possible to serve a valid JSON-LD output defining and validating a JSON-LD template through its UI. It also proved to be very flexible in providing support for templating various output formats and building simple or complex content negotiation rules. The capability it provides in transforming flat data structures (simple features) to complex nested ones as a JSON-LD has been fundamental.

Some ways to improve the Features Templating plug-in were identified.

- Its UI is text based and the templates can only be generated by direct text editing. It could be improved by providing a UI tool that allows a graphical mapping among source properties and template properties, making the template generation even more easy to use.
- Templates make use of vendor option definitions in order to allow the customization of the content outside of the feature's scope: the `@context` itself needs to be provided inside an JSON object `$options`. It could be improved by removing the necessity of using $options, thus directly writing the template parts that are not mapping feature attributes as they will actually result in the final output, according to a pure what you see is what you get (WYSIWYG) approach.

9.2. Finding Appropriate Vocabularies

Mapping some of the terms has been more challenging than expected for two main reasons.

- The selected schema.org type seems meaningful for the use case of this Pilot but the type properties don't semantically overlap completely with the ones found in the test dataset.

This has been the case of the TRI dataset layer for which the `SpecialAnnouncement` type was used.

The selected schema.org type semantically fits the dataset but the concept it expresses is too generic. This has been the case of the MRI dataset layer for which the more generic type `Dataset` was chosen.

- Due a lack of schema.org types that have properties of types able to represent geospatial terms, it was necessary to use the `FeatureCollection` term mapped to an xsd IRI in order to give a `@type` to the root JSON object.

  Schema.org provides useful geospatial types like `Place`, `AdministrativeArea`, and `TouristAttraction`, that cover various kinds of Geospatial locations. On the other hand, these types are not used as properties in other schema.org types. This causes some problems in the applicability to georeferenced data of various terms.

9.3. Limitation to Static Pages

Publishing the HTML Feature output posed an additional problem: Google accepts only HTML documents identified by a URL without any additional parameter for indexing . On the contrary, any `GetFeature/GetFeatureInfo` operation performed on a GeoServer instance must include within the URL several query parameters such as the service, the operation, the `featureType` being requested, and the output format, to cite few. This forced GeoSolutions to create and publish a static HTML page with the encoded `FeatureCollection` and the JSON-LD document.

9.4. Duplicated Data

When GeoServer injects the JSON-LD document inside an HTML output of a feature collection, the feature collection needs to be encoded two times (one to obtain the HTML representation and one for the JSON-LD). This means that the impact over the response time is significant.

Search engines in general need the whole JSON-LD document to be embedded inside the HTML page (see here, for example). If it is provided differently, e.g., as an external ref, they will ignore it.

10

RECOMMENDATIONS

10 RECOMMENDATIONS

This chapter includes recommendations for future work. Future work should build upon the findings that emerged from the development and testing of these components and answer questions that were out of scope.

10.1. Adding JSON-LD Support to Metadata

Since the HTML publishing process does not support URLs with query strings, any possibility of dynamic filtering of the data being provided in the HTML page was lost. Despite improving the indexability of geographic data, using static HTML pages does not seem feasible since WebGIS are mainly based on dynamic web map content.

OGC API — Features provides a starting point to help solve this issue. Currently, JSON-LD output format is supported only when retrieving the data (the collections items). By expanding this support to the retrieval of metadata, it could be then possible to improve the indexability of GIS websites.

To support this scenario, embedding JSON-LD should be considered to:

- `/ogc/features` output available as a starting point to navigate the API;
- `/ogc/features/collections` output where a list of available collections (set of data) is made available; and,
- `/ogc/features/collections/{collectionName}` output where details about a single collection are provided.

Given that support, any frontend application used to set up a WebGIS could then reference on the home page, the `/ogc/features` endpoint, to give a starting point for search engines to start the crawling of information about the content of the application.

The link to the OCG-API — Features starting point should not be advertised to the user and could then be provided by the front-end, either as a hidden anchor element or as link inside the JSON-LD document itself, although this last option is not currently support by search engines nor the JSON-LD standard itself.

10.2. Exploring the use of JSON-LD Provided Externally

One suggestion to overcome the issue of having data duplicated in both the `@context` and the HTML body would be to have search engines support JSON-LD as an external ref. This cannot be done through the script element since, according to the specification, when using it as a data

element (which is the case of JSON-LD) the `src` attribute needs to be empty. But the same could be achieved by using the link element, e.g.,

```
<link href="https://here/where/retrieve/jsonld.js" rel="alternate" type=
"application/ld+json" />
```

Figure 21

10.3. Develop Vocabulary Types with Geospatial Properties

Based on the difficulties that emerged during the process of choosing a vocabulary for the JSON-LD example datasets, future developments in schema.org might consider defining terms that better fit the structure of collections of geospatial data served in JSON formats, which normally consist in a root JSON object with an array attribute populated with several feature objects, each one with its own geometry property. In general, it would be useful that in schema.org, types with geospatial properties appear as instances for properties of other terms. This would overcome problems like the one faced with the `SpecialAnnounement` type.

10.4. Explore the Usage of GLNs with Linked Data

Future work could see continued and actual use of GS1 GLN to identify disaster related facility and location information that can assist with efficient routes as well as be the key to disaster related facility and location information via GS1 Digital Link. Its combination with Linked Data could boost the discoverability of disaster-related information by enabling the search of such information using GLNs. This type of search could return disaster-related information associated with a specific area or facility identified by a GLN.

A

ANNEX A (INFORMATIVE) REVISION HISTORY

A ANNEX A (INFORMATIVE) REVISION HISTORY

DATE	RELEASE	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
October 26, 2021	.1	S. Taleisnik	all	Initial version
February 13, 2022	.3	S. Taleisnik	all	Preliminary draft
May 25, 2022	.8	S. Taleisnik	all	Final draft
June 27, 2022	.9	J. Lieberman	all	Final review
June 28, 2022	1	S. Taleisnik	all	Final version

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] OGC Disaster Pilot 2021: Call for Participation (CFP) (2021). https://portal.ogc.org/files/?artifact_id=97381

[2] Semantic Web (2015). https://www.w3.org/standards/semanticweb/

[3] An Introduction to the Global Location Number (GLN) (2017). https://www.gs1us.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=158&language=en-US&PortalId=0&TabId=134

[4] Understand how structured data works (2021). https://developers.google.com/search/docs/advanced/structured-data/intro-structured-data

[5] About Schema.org (2021). https://schema.org/

[6] OGC Disaster Pilot Summary Engineering Report (2022).