Open Geospatial Consortium

Publication Date: 2015-11-18

Approval Date: 2015-08-27

Posted Date: 2015-07-21

Reference number of this document: OGC 15-058

Reference URL for this document: http://www.opengis.net/doc/PER/tb11-symbology-mediation

Category: Public Engineering Report

Editor(s): Stephane Fellah

OGC Testbed-11 Symbology Mediation Engineering Report

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

Document type:	OGC® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

License Agreement

Contents

Page

Figures

Page

Tables

Page

Abstract

This OGC® Engineering Report (ER) summarizes the approaches, findings and the results of the Symbology Mediation sub-thread activities of the OGC Testbed-11 Cross Community Interoperability (CCI) Thread. The ER:

- Provides an overview of existing standards relevant to symbology mediation,
- Outlines the approaches adopted during the testbed,
- Describes the conceptual models and services developed during the testbed to address semantic mediation and portrayal of feature information related to Emergency Management and to some extent to the Aviation domain.

Business Value

This Engineering Report proposes a solution to improve semantic interoperability in the following areas:

- Incident management in the context of Emergency Management, Law Enforcement and Public Safety.
- Semantic mediation
- Portrayal of features using Linked Data standards.

Keywords

ogcdocs, testbed-11, ogcdoc, ogc documents, ows11, ontology, cci, GeoSPARQL, portrayal, symbology, mediation, alignment, semantic, UML, owl, incident mapping, emergency.

Testbed-11 Symbology Mediation

1 Introduction

1.1 Scope

This Engineering Report (ER) discusses the implementation of the models and services associated to perform semantic mediation and symbology portrayal. The report also discusses interoperability and standards gaps identified during the testbed and provides recommendations for future work.

1.2 Document contributor contact points

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

Name	Organization
Stephane Fellah	Image Matters LLC
Gobe Hobona	Envitia
Frederick Houbie	Geomatys

1.3 Future work

For recommendations on future work please refer to section 14.

1.4 Foreword

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.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC 05-078r4, *OGC® Styled Layer Descriptor Profile of the Web Map Service Implementation Specification*

OGC 05-077r4, *OpenGIS Symbology Encoding Implementation Specification*

OGC 07-147r2, *OGC KML*

ISO 19117:2012, *Geographic Information – Portrayal*

OGC 11-052r4, *OGC GeoSPARQL –A Geographic Query Language for RDF Data*

OGC 05-110, *Feature Portrayal Service*

OGC 11-063r6, *OWS-8 CCI Semantic Mediation Engineering Report*

OGC 12-103r3, *OWS-9 CCI Semantic Mediation Engineering Report*

OGC 14-049, *Testbed 10 OWS CCI Ontology Engineering Report*

OGC 14-106. *Unified Geo-data Reference Model for Law Enforcement and Public Safety*

OGC-15-054 *Implementing Linked Data and Semantically Enabling OGC Services Engineering Report*

In addition to this document, this report includes several OWL Ontology Document files referred in Annex A and Annex B.

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

3.1
feature
representation of some real world object or phenomenon

3.2
interoperability
capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units [ISO 19119]

3.3
map
pictorial representation of geographic data

3.4
model
abstraction of some aspects of a universe of discourse [ISO 19109]

3.5
ontology
a formal specification of concrete or abstract things, and the relationships among them, in a prescribed domain of knowledge [ISO/IEC 19763]

3.6
portrayal
portrayal presentation of information to humans [ISO 19117]

3.7
semantic interoperability
the aspect of interoperability that assures that the content is understood in the same way in both systems, including by those humans interacting with the systems in a given context

3.8
semantic mediation
transformation from one or more datasets into a dataset based on a different conceptual model.

3.9
symbol
a bitmap or vector image that is used to indicate an object or a particular property on a map.

3.10
symbology encoding
style description to apply to the digital features being rendered

3.11
syntactic interoperability
the aspect of interoperability that assures that there is a technical connection, i.e. that the data can be transferred between systems

4 Conventions

4.1 Abbreviated terms

CAP	Common Alert Protocol
CCI	Cross Community Interoperability
E&DM	Emergency and Disaster Management
EDXL	Emergency Data Exchange Language
ER	Engineering Report
FPS	Feature Portrayal Service
GML	Geography Markup Language
HTML	HyperText Markup Language
IMS	Incident Management System
JSON-LD	JavaScript Object Notation for Linked Data
LEAPS	Law Enforcement and Public Safety
NIEM	National Information Exchange Model
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
OWS-8	OGC Web Services Initiative, Phase 8
OWS-9	OGC Web Services Initiative, Phase 9
RDF	Resource Description Framework
SDI	Spatial Data Infrastructure
SE	Symbology Encoding
SLD	Style Layer Descriptor
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language

SVG	Scalable Vector Graphics
URI	Unique Resource Identifier
URL	Uniform Resource Locator
URL	Uniform Resource Name
WFS	Web Feature Service
WKT	Well Known Text
WMS	Web Map Service

4.2 UML notation

Some diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3].

5 ER Topic overview

Incident management plays a crucial role in many application domains including Homeland Security, Law Enforcement and Public Safety (LEAPS), and Emergency and Disaster Management (E&DM). An Incident Management System (IMS) needs to facilitate rapid, agile and effective engagement of first responders at national, state and local jurisdictional levels to respond to emergency incidents that may pose an immediate security threat to human life and/or the flow of commerce. The current systems face the following challenges.

- Analysts and operators need to quickly triage, fuse, connect dots, detect patterns, infer insights, and make sense of the flow of incident information to get an unambiguous Common Operational Picture using symbology that makes sense to the users.
- Users need to integrate and interpret incidents, observations, mutual aid requests, alerts, symbologies, taxonomies, etc. generated across a multi-agency, multi-jurisdictional spectrum.
- There is limited interoperability between agencies due to different protocols, taxonomies, models and symbolic representations (stovepipes are still there!).

The current data-centric implementations are mainly based on syntactic and structural approaches and thus imposes a huge cognitive burden on the users to make sense of the large volume and varieties of information.

Data model standardization relies upon homogeneous data description and organization. This imposes strict adherence to a standard that is defined at the syntactic-schematic level. Achieving consensus in definition of the data model is difficult and the final model is less flexible. Modelers struggle between producing simple models where it is easier to gain consensus but harder to achieve desired business reality versus those seeking richer models that are closer to reality but have unwanted complexity.

Data-centric approaches (using for example XML Schema, or JSON) increase the chance for multiple interpretations and misinterpretations of data. Data interpretation requires knowledge of the semantics (e.g., meanings, significance, relevance, etc.) and surrounding context. Data-centric approaches are unable to capture these semantics and context, which are in turn required for automated fusion, analytics, and reasoning.

To address these challenges in the domain of symbology mediation related to E&DM, we adopted a knowledge-centric approach. The knowledge-based approach employs a standards-based formal, sharable framework (RDF, OWL, SPARQL) that provides a conceptual domain model to accommodate various business needs. Among these standards, there are:

- Resource Description Framework (RDF) which provides the core model for representing data in the form of a semantic graph composed of triples (W3C standard);

- RDF Schema which provides a simple vocabulary to define classes and properties (W3C standard);

- Web Ontology Language (OWL) which provides a more expressive vocabulary to define ontologies than RDF Schema (W3C standard); and

- SPARQL Protocol and RDF Query Language (SPARQL) defines a query language for RDF data, analogous to the Structured Query Language (SQL) for relational databases (W3C standard).

Decentralized model extensions can be accommodated without adversely affecting the existing information infrastructure. Using these standards, a set of ontologies were developed during this testbed to represent incidents, portrayal information and semantic mediation mappings.

This ER is structured the following way.

- Provides an overview about symbologies in E&DM, with a focus on two particular symbologies that were used in this testbed: the FGDC HSWG Emergency Management Symbology and the Canadian Emergency Management Symbology.

- Provide a review of the existing standards related to symbology that were taken in account in the design of the portrayal ontologies.

- A review of incident models.

- Describe in detail the Portrayal Ontologies designed and implemented during this testbed. We show how we encoded the HSWG EMS and Canadian EMS symbology using these ontologies.

- Introduce ontologies to support semantic mediation. The first ontology introduces a number of extensions to SPARQL to represent functions, rules, templates, queries in a Linked Data form. The second ontology is used to define semantic alignment between two ontologies.

- Illustrate how these ontologies are used to implement semantic mediation between two simple incident models.

- Document the different services implemented and datasets used by the different participants of this thread, in particular the semantic mediation service and the semantic portrayal catalog service.

We end the ER document with a summary of challenges encountered during the testbed and the recommendations for future work.

6 Symbology in Emergency Management

6.1 Overview

A plethora of symbols and symbol schemes exist. However, despite the large number of initiatives targeting exactly such a standardized set, very few standardized sets can be found. Many of those initiatives date back to the first decade 2000-2009 and have failed to become widely accepted and applied mapping symbol standards.

There are various Symbology Best Practices available for Law Enforcement and Public Safety (LEAPS) used in the context of Emergency Management.

- US FGDC ANSI 415-2006 INCITS Homeland Security Map Symbol Standard
- Canadian Emergency Management Symbology (EMS)
- United Nations Office of Coordination on Humanitarian Affairs (UN-OCHA), which has created a set of 500 freely available humanitarian icons to help relief workers present emergency and crisis-related information quickly and simply
- Portuguese Disaster Response Map Symbols (DRMS) project, an effort to create a standard set of symbols that may aid disaster managers and responders to create efficient maps
- Australasian All-Hazards Symbology, developed by the Intergovernmental Committee on Surveying and Mapping (ICSM) and the Victoria-based company Spatial Vision
- The European INDIGO project Emergency 2D/3D Symbology Reference
- World Meteorological Organization (WMO) Intergovernmental Oceanographic Commission efforts on map symbol standardization

Since in house-symbols are preferred, few of these best practices have been widely adopted. In the case of a multi-institutional activities the lack of a common set of symbology can be problematic. Semantic Mediation of Symbology provides a viable solution that allows the representation of an incident information using symbology of a given community to a target incident information model using symbology of another community.

6.2 FGDC HSWG Emergency Management Symbology

The US Federal Geographic Data Committee Homeland Security Working Group (FGDC HSWG) symbology set, standardized by the American National Standards Institute (ANSI), divides symbols into four categories:

- **Incidents**: cause of action or source of disaster;
- **Natural events**: phenomenon created by naturally occurring conditions;
- **Infrastructure**: basic facilities, services and installations needed for the functioning of a community; and
- **Operations**: capabilities or resources available during or implemented due to an emergency.

Borders and patterns around these shapes are used to visually classify the symbols into their respective groups. The FGDC HSWG Emergency Management Symbology is designed in black and white and uses TrueType fonts for the icons.

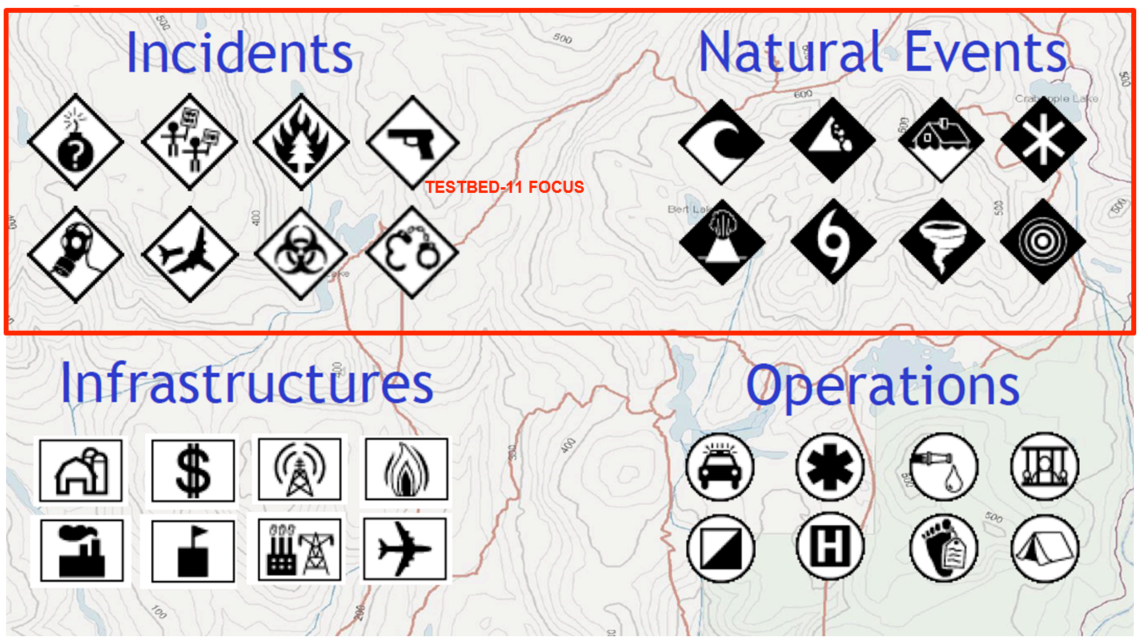


Figure 1: HSWG Emergency Symbology Samples

For this testbed, we encoded the natural events and incidents using Simple Knowledge Organization System (SKOS) encoding and then aligned the encoding of the symbols with the Symbology Ontology produced during this testbed. We leverage some of the work performed during the OGC Testbed 10 (OGC 14-049).

6.3 Canadian Emergency Management Symbology

The Canadian Emergency Mapping Symbology (EMS) is designed to be used in both single and multi-agency emergency mapping applications to facilitate interoperability and situational awareness. The user community consists of federal, provincial, regional and local organizations involved in the management of major events, disasters, and other incidents where emergency help and security are needed. From a command and control perspective, which includes the military and many civilian organizations, such occurrences are referred to as *incidents*. Some civilian agencies though have preferred the term *event* instead. As used by EMS, the terms incident and event are interchangeable. In addition to incidents, an understanding of the overall picture describing an emergency requires knowledge of infrastructures and operations. These terms are widely accepted, and consequently are used in EMS as well.

A symbology includes a set of symbol definition. As important, symbology also includes a classification of the entities under consideration. A four level, hierarchical taxonomy is used here. At the highest level, all entities fall in the EMS domain. *Incident*, *infrastructure* and *operation* are considered as categories within that domain, as are *aggregate* and *other*. Other domains may exist with other categories. Each category is subdivided further to form a set of Tier 1 classes; each of these in turn is broken down further to create Tier 2 classes. A diagram showing the structure of the classification is shown in Figure 2: EMS Classification Structure

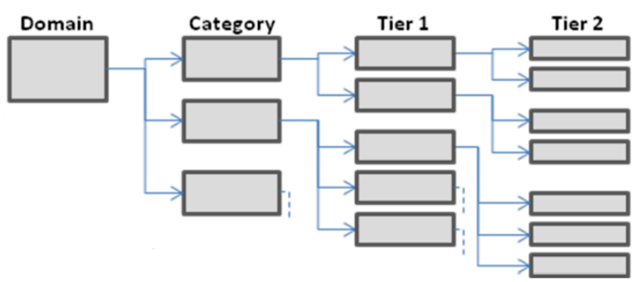


Figure 2: EMS Classification Structure

The classification can also be represented in text by a simple dot notation:

Tier 1 entity: **domain.category.tier1**

Tier 2 entity: **domain.category.tier1.tier2**

Aviation	`ems.incident.aviation`	
Aircraft Crash	`ems.incident.aviation.aircraftCrash`	
Definition: A sudden, unexpected event involving aircraft resulting in fuselage damage, bodily injury, death and/or the disruption of transportation service; prompting emergency landing procedures or uncontrolled impact with the ground.		
Aircraft Hijacking	`ems.incident.aviation.aircraftHijacking`	
Definition: The unexpected, unlawful and forceful seizure of control aboard an aircraft by an individual or group of individuals resulting in passenger and crew endangerment, injury or death, and/or the redirection of flight destination.		
Airport Closure	`ems.incident.aviation.airportClosure`	
Definition: A closure of an airport or helicopter landing site.		
Airspace Closure	`ems.incident.aviation.airspaceClosure`	
Definition: A closure of a jurisdiction's airspace to air traffic.		
Notice to Airmen	`ems.incident.aviation.noticeToAirmen`	
Definition: A Notice to Airmen (NOTAM) is created and transmitted by a government agency under guidelines specified by Annex 15: Aeronautical Information Services of the Convention on International Civil Aviation. A NOTAM is filed with an aviation authority to alert aircraft pilots of any hazards en route or at a specific location.		

GeoConnections *Emergency Mapping Symbology v1.0* *Page 24*

Figure 3: Canadian EMS Symbols and taxonomy

For this project, we manually encoded the EMS Incident taxonomy using SKOS encoding. We then performed a manual mapping from the EMS concepts to HSWG concepts using SKOS mapping properties (skos:exactMatch, skos:closeMatch, skos:broaderMatch,…).

Here a sample of the mapping

@prefix : <http://www.opengis.net/taxonomy/ems-hswg_mapping#> .
@prefix ems: <http://www.opengis.net/taxonomy/ems#> .
@prefix incidents: <http://www.fgdc.gov/HSWG/taxonomy/incidents#> .
@prefix natural-events: <http://www.fgdc.gov/HSWG/taxonomy/natural-events#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix spin: <http://spinrdf.org/spin#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.opengis.net/taxonomy/ems#ems.incident>
skos:exactMatch incidents:Incident ;

.
<http://www.opengis.net/taxonomy/ems#ems.incident.airQuality>
skos:broadMatch incidents:Incident ;
.
<http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilDemonstration>
skos:exactMatch incidents:CivilDemonstrations ;
.
<http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilDisplacedPopulation>
skos:exactMatch incidents:CivilDisplacedPopulation ;
.
<http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilEmergency>
skos:broadMatch incidents:CivilDisturbanceIncident ;
.
<http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilRioting>
skos:exactMatch incidents:CivilRioting ;
..
<http://www.opengis.net/taxonomy/ems#ems.incident.animalHealth.animalFeed>
skos:broadMatch incidents:Incident ;
.
<http://www.opengis.net/taxonomy/ems#ems.incident.aviation>
skos:exactMatch incidents:AirIncident ;
.
<http://www.opengis.net/taxonomy/ems#ems.incident.aviation.aircraftCrash>
skos:broadMatch incidents:AirAccident ;
.

7 Review of existing Portrayal standards

A number of existing standards related to Portrayal were considered during this testbed to help define an ontology.

7.1 ISO 19117

ISO 19117:2012 specifies a conceptual schema for describing symbols, portrayal functions that map geospatial features to symbols, and the collection of symbols and portrayal functions into portrayal catalogues. This conceptual schema can be used in the design of portrayal systems. 19117 allows feature data to be separate from portrayal data, permitting data to be portrayed in a dataset independent manner.

7.2 SLD

The OpenGIS® Styled Layer Descriptor (SLD) Profile of the OpenGIS® Web Map Service (WMS) Encoding Standard [http://www.opengeospatial.org/standards/wms] defines an encoding that extends the WMS standard to allow user-defined symbolization and coloring of geographic feature and coverage data. SLD addresses the need for users and software to be able to control the visual portrayal of the geospatial data. The ability to define styling rules requires a styling language that the client and server can both understand. The OpenGIS® Symbology Encoding Standard (SE) provides this language, while the SLD profile of WMS enables application of SE to WMS layers using extensions of WMS operations. Additionally, SLD defines an operation for standardized access to legend symbols.

7.3 SE

This OGC standard defines Symbology Encoding, an XML language for styling information that can be applied to digital Feature and Coverage data. This document is together with the Styled Layer Descriptor Profile for the Web Map Service Implementation Specification the direct follow-up of Styled Layer Descriptor Implementation Specification 1.0.0. The old specification document was split up into two documents to allow the parts that are not specific to WMS to be reused by other service specifications.

7.4 KML

KML is an XML language focused on geographic visualization, including annotation of maps and images. Geographic visualization includes not only the presentation of graphical data on the globe, but also the control of the user's navigation in the sense of where to go and where to look.

From this perspective, KML is complementary to most of the key existing OGC standards including GML (Geography Markup Language), WFS (Web Feature Service) and WMS (Web Map Service). Currently, KML 2.2 utilizes certain geometry elements derived from GML 2.1.2. These elements include point, line string, linear ring, and polygon.

8 Incident Ontologies

8.1 ADP LEAPS Model

In 2014, the OGC published a best practice document providing an overview of the Unified Geo-data Reference Model for Law Enforcement and Public Safety (Unified Model) [OGC 14-106]. The Unified Model was originally developed by the GIS Center for Security (GIS CS), Abu Dhabi Police. The GIS CS was initiated based on a UAE Ministry of Interior issued decree to establish GIS CS with the core mission: “To geo-enable police services and applications using International standards and best practices.” In 2010, the GIS SC initiated a program to develop a Standardized GIS Environment (SGA). Part of this effort was to define and implement a standard data model for sharing Law Enforcement and Public Safety data.

The reference model development effort also included other organizations. While any given Law Enforcement organization will have differences in how it models the real world, there are many aspects of the domain such as incident location that are common across organizations. However, currently there is no data model best practice for the Law Enforcement community to use as a reference for these common information elements. The Reference Model described in the referenced diagrams in the document is provided as an exemplar and a starting point to help facilitate greater commonality and interoperability between organizations. As it is common within this law enforcement domain, policing agencies will not utilize any reference model until it has been deemed a best practice. Upon acceptance the expectation is that modifications and enhancements will occur to meet the needs of a wider, more global population.

This best practice is not meant to compete or replace existing standards such as the NIEM (National Information Exchange Model) standard. The NIEM emphasizes data exchange between organizations typically within the US, and is based on XML, which reflects only a small subset of the information of interest to a full data model.

For this testbed, we analyzed only a subset of the unified reference model that relates to incident modeling and took it in account to refine the core Incident Ontology initiated during the OGC testbed 10. Overall the Incident ontology proposed in OWS-10 was consistent with the incident model of Unified Reference Model for LEAPS and could easily accommodate the specificities related to LEAPS by extending the proposed core Incident Ontology described in the next section.

Based on the analysis, we concluded that the unified reference model for LEAPS can be used to derive a set of foundational ontologies that serve as the basis for interoperability throughout LEAPS domain. These ontologies should be built upon a set of core OGC's Geospatial Ontologies and take also in account other standards such as NIEM, EDXL and CAP. They should define the minimum-essential core set of concepts, properties and relationships for the LEAPS community. The LEAPS ontologies must be designed to accommodate any jurisdiction/mission profile specific information (taxonomies, specialized concepts and relationships), using the built-in extension mechanisms of OWL.

8.2 Core Incident Model

For OGC Testbeds 10 and 11, we investigated the formalization of a core ontology for representing incident information used by IMS. The goal of this ontology is to provide a semantic model for Incident Management systems that could be adapted across multiples domains and jurisdictions to enable interoperability of incident management systems between these domains. The Incident ontology attempts to capture the minimal set of concepts and properties that are truly cross-domain. Using the built-in extensibility and multilingual mechanisms of OWL and SKOS, the Incident ontology can be adapted to build specific profiles for Military and Civilian IMS while still being interoperable because they are based on the same core concepts. Our Incident ontology is built-on a set of core ontologies (spatial, temporal, identifier, event, SKOS) to provide a coherent model for geospatial reasoning on incidents.

Figure 4 gives an overview of the Incident Ontology Model. Incidents are based on the Event ontology defined in the Core Geospatial Ontologies.

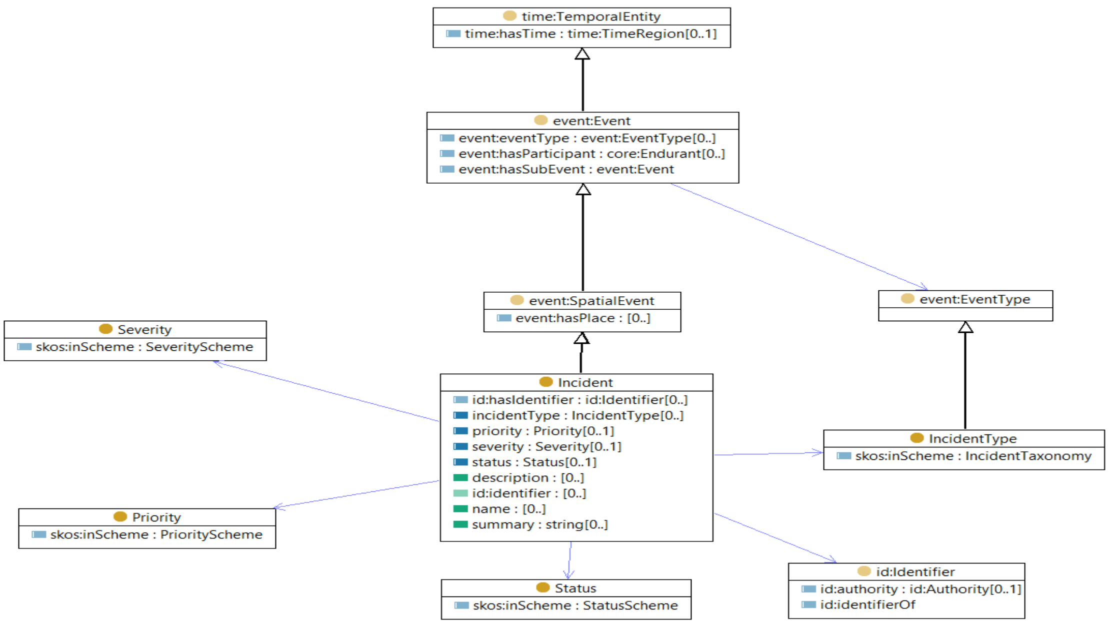


Figure 4: Core incident ontology model

The core Incident Model can be extended using the built-in mechanism in OWL to add specific information used within an application domain or jurisdiction, thus becoming specialized profile of the core Incident model. Existing standards based on syntactic and schematic standards (NIEM, EDXL, NDEX,…) can be integrated in Semantic Incident Management Systems by the use of adapters converting the standard to an RDF representation using the different incident profiles (see Figure 5). The benefit of this approach is to provide a unified semantic representation of incidents that can integrate different systems without breaking incompatibilities. The knowledge-based model allows machines to interpret information without ambiguities, perform triaging, reasoning and fusion of information. The system will assist the user by reducing the cognitive burden to make sense of the information. The common knowledge-based representation of the incidents can be used to convert back the information to existing data-centric standard. The semantic layer introduces a layer of 'smartness' in the existing IMS that can adapt easily to future change in the model.

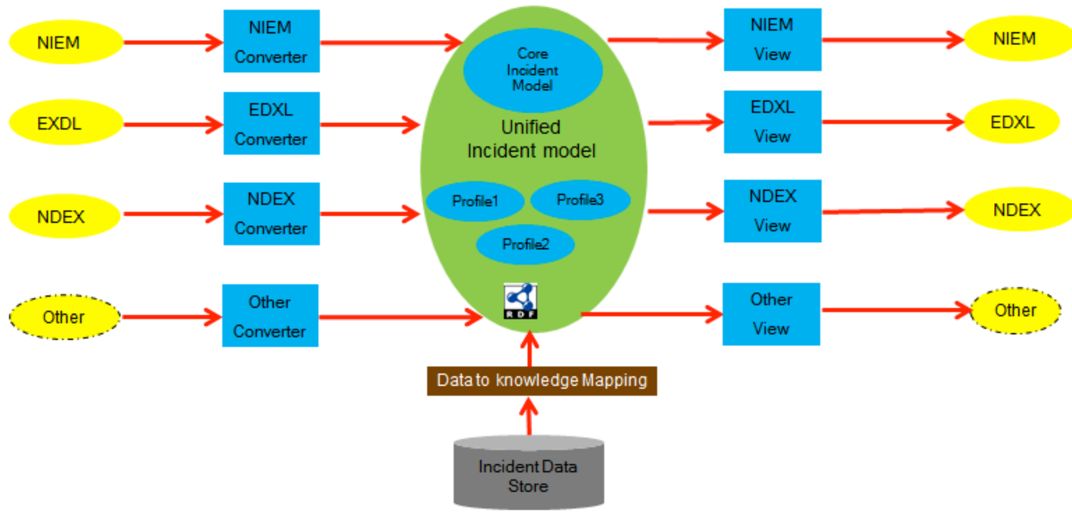


Figure 5: Semantic layer with adapters to the core incident model and derived profiles

8.3 Incident model and data for demonstration

During the Testbed, we were unable to get datasets for incidents at national level that use HSWG or Canadian EMS taxonomies and could exercise our core incident ontology. We recommend strongly that in future Testbeds, datasets are provided by sponsors or OGC at the start of the testbeds to be able to run scenarios. As a fallback, close to the end of the testbed, we used open-source data from the San Francisco Police Department (SFPD) (https://data.sfgov.org/Public-Safety/SFPD-Incidents-from-1-January-2003/tmnf-yvry), as the demonstration was occurring in the San Francisco Bay.

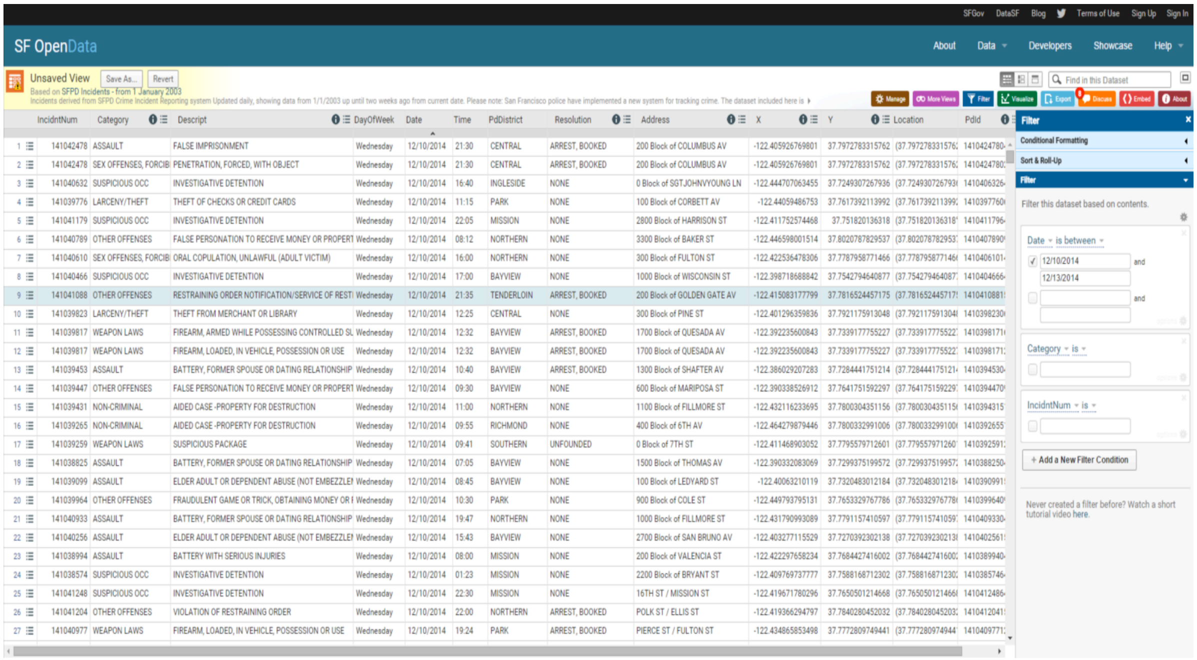


Figure 6: SFPD Incident samples from SF OpenData

The data were exported in CSV format and the values of category of the incident were replaced with concepts from the HSWG EMS taxonomy in order to leverage the SKOS encoding of HSWG. We defined intentionally two different incident models to exercise the semantic mediation of taxonomies and ontologies.

Here a sample of a HSWG Incident using the HSWG EMS Taxonomy.

```
@prefix ks:    <http://www.usersmarts.com/ont/2005/06/ks#> .
@prefix spatial: <http://www.opengis.net/ont/spatial#> .
@prefix hswg:  <http://www.opengis.net/testbed11/ont/incident/hswg#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix time:  <http://www.knowledgesmarts.com/ontology/time#> .
@prefix evt:   <http://www.knowledgesmarts.com/ontologies/event#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix place: <http://www.knowledgesmarts.com/ontologies/place#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix evt-type: <http://www.smartrealm.com/ct/types/events#> .
@prefix ptype: <http://www.knowledgesmarts.com/ontologies/place/types#> .
@prefix incident: <http://www.opengis.net/ont/domain/emergency/police/incident#> .

<http://ows.usersmarts.com/ldapp/ows11/demo/ems/sfpd/incidents/11616059406244>
    a                   hswg:HSWGIncident ;
    hswg:hasAddress     [ a                 hswg:Address ;
```



```
                    hswg:city           "San Francisco" ;
                    hswg:fullAddress     "SHERIDAN ST / 9TH ST" ;
                    hswg:policeDistrict  "SOUTHERN" ;
                    hswg:state           "CA"
                  ] ;
    hswg:incidentDate    "2011-12-11"^^xsd:date ;
    hswg:incidentNumber  "116160594" ;
    hswg:incidentTime    "03:00:00"^^xsd:time ;
    hswg:incidentType    <http://www.fgdc.gov/HSWG/taxonomy/incidents#Looting>;
    hswg:location        [ a              wgs84:Point , geosparql:Point ;
        geosparql:asWKT  "POINT (-122.4106935 37.77302471)"^^geosparql:wktLiteral ;
        wgs84:lat        37.77302471 ;
         wgs84:long      -122.4106935
      ] ;
    hswg:resolution      "NONE" ;
    hswg:summary         "GRAND THEFT FROM LOCKED AUTO" .
```

This same incident is expressed using a different Incident Model and taxonomy for Canadian EMS

```
@base          <http://www.opengis.net/taxonomy/ems#> .
@prefix ogc-map: <http://www.opengis.net/testbed11/ont/geosparql/mapping/core#> .
@prefix sd:    <http://www.w3.org/ns/sparql-service-description#> .
@prefix natural-events: <http://www.fgdc.gov/HSWG/taxonomy/natural-events#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix ems:   <http://www.opengis.net/taxonomy/ems#> .
@prefix lda:   <http://www.knowledgesmarts.com/ontologies/lda#> .
@prefix incidents: <http://www.fgdc.gov/HSWG/taxonomy/incidents#> .
@prefix geosparql-fn: <http://www.opengis.net/testbed/11/def/function/geosparql/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix mediation: <http://www.opengis.net/testbed11/ont/geosparql/mediation#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix spin:  <http://spinrdf.org/spin#> .
@prefix sparql-ext: <http://www.opengis.net/testbed11/ont/geosparql/extensions#> .
@prefix fn:    <http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix incident: <http://www.opengis.net/ont/emergency/incident#> .
```

```
<http://ows.usersmarts.com/testbed11/data/ems#11616059406244>
      a       <http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident> ;
      <http://www.opengis.net/testbed11/ont/incident/ems#address>
            [ a                vcard:Address ;
              vcard:locality       "San Francisco" ;
              vcard:region         "CA" ;
              vcard:street-address  "SHERIDAN ST / 9TH ST"
            ] ;
      <http://www.opengis.net/testbed11/ont/incident/ems#description>
            "GRAND THEFT FROM LOCKED AUTO" ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentDate>
            "2011-12-11"^^xsd:date ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentId>
            "116160594" ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentTime>
            "03:00:00"^^xsd:time ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentType>
            ems:ems.incident.crime.looting ;
      <http://www.opengis.net/testbed11/ont/incident/ems#position>
            [ a              geosparql:Point ;
              geosparql:asWKT  "POINT (-122.4106935
37.77302471)"^^geosparql:wktLiteral ;
              wgs84:lat       37.77302471 ;
              wgs84:long      -122.4106935
            ] .
```

The SFPD linked data model was made available at the following endpoint:

http://ows.usersmarts.com/ldapp/ows11/demo/ems/sfpd

The goal of the semantic mediation is to transform the HSWG Incident to an EMS Incident instance and mapping the HSWG taxonomy to the Canadian EMS taxonomy. The Semantic Mediation section of this document will demonstrate how this has been accomplished by using a semantic mapping description between both ontologies and taxonomies expressed as Linked Data.

9 Portrayal Ontologies

9.1 Overview

The Portrayal Ontologies specify a conceptual model for portrayal data, in particular symbols and portrayal rules. Portrayal rules associate features with symbols for the portrayal of the features on maps and other display media. These ontologies include classes, attributes and associations that provide a common conceptual framework that specifies the structure of and interrelationships between feature types, portrayal rules and symbols. It separates the content of the data from the portrayal of that data to allow the data to be portrayed in a manner independent of the dataset. The graphic description is intended to be format independent but convertible to any target formats (SVG, KML). The ontologies are derived from concepts found in existing portrayal specifications (ISO 19117, OGC Symbology Encoding and Styled Layer Descriptor Profile of WMS).

We define microtheory as a ontology containing a small set of concepts, attributes and relationships that are consistent with a given theory. To favor reusability, the Portrayal ontologies are decomposed into four microtheories (see Figure 7).

- **Style ontology**: defines the concept of Style and portrayal rules.
- **Symbol ontology**: defines the concept of Symbol Set and Symbol and structural definition of Symbol components.
- **Graphic Ontology**: defines graphic elements including graphic objects and attributes.
- **Portrayal Catalog Ontology**: Defines the concept of Portrayal Catalog

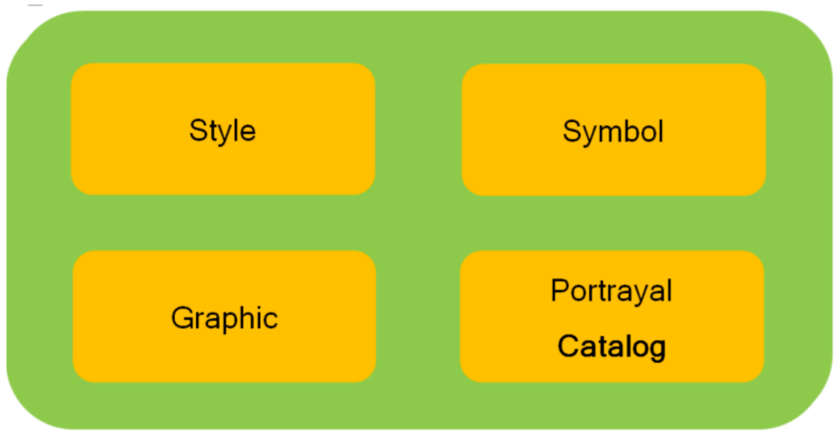


Figure 7: Portrayal Microtheories

The scope of the testbed was to mainly focus on point-based symbology to represent emergency management incidents. More complex symbology such as line and area-based symbol has been intentionally left out of the ontologies as future extensions of the model, due to the limited timeframe for this testbed. However, we designed the ontologies in such a way that they can easily accommodate future extensions to represent more complex symbology. Our intent was to lay out a foundational framework for portrayal ontologies that will allow future extensions.

The details of each microtheory are described in the following sections. The namespace mapping for each microtheory is defined in Table 1 Namespace mapping for Portrayal Microtheories

Table 1 Namespace mapping for Portrayal Microtheories

Microtheory	**Namespace**	**Prefix**
Style	http://www.opengis.net/ont/portrayal/style#	Style
Symbol	http://www.opengis.net/ont/portrayal/symbol#	symbol
Graphics	http://www.opengis.net/ont/portrayal/symbol#	graphics
Portrayal Catalog	http://www.opengis.net/ont/portrayal/catalog#	catalog

9.2 Design Approach

This section outlines some the key principles used to design the portrayal ontologies.

9.2.1 Minimal ontological commitment

Our modular design of the ontologies follows the principle of making a minimal ontological commitment to the nature of concepts and of relationships between concepts. As explained by Thomas Gruber [4] an ontology should require the minimal ontological commitment sufficient to support the intended knowledge sharing activities. An ontology should make as few claims as possible about the world being modeled, allowing the parties committed to the ontology freedom to specialize and instantiate the ontology as needed (which is often called ontology profile).

Opting for such a minimal approach is made dramatically easier by the vocabulary extension mechanisms offered natively by Semantic Web technology. Applications that require more constrained behavior may define compatible extensions to OWL or SKOS. For example, modelers may coin sub-classes and sub-properties of OWL or SKOS properties, or associate those properties with specific formal axioms. By making a minimal ontological commitment, the ontologies can be applied and reused across multiple Communities of Interests (COIs), thus increasing the rate of wide-spread adoption.

9.2.2 Modularization of ontologies

Quoting Stuckenschmidt and Klein [5], "ontologies that contain thousands of concepts cannot be created and maintained by a single person." Modularization helps designers manage complexity by reducing the size of the design problem [6]. We want designers design modules of a size that they can apprehend, and later either integrate these modules into a final repository or build the relationships among modules that support interoperability. This is a typical application of the divide-and-conquer principle.

Modularization also provides a way to keep performance of ontology services at an acceptable level. Performance concerns may be related to query processing techniques, reasoning engines and ontology modeling and visualization tools. Reasoners currently available are performing well on small-scale ontologies, with performance degrading rapidly as the size of the ontology increases. Keeping ontologies small is one way to avoid the performance loss, and modularization is a way to replace an ontology that tends to become oversized by smaller subsets. Modularization fulfills the performance goal if, whenever a query has to be evaluated or an inference to performed, this can be done by looking at just few modules, rather than exploring the whole ontology (Stuckenschmidt et al., 2009).

9.2.3 Reusability of ontologies

Reusability is a well-known goal in software engineering. Reuse is most naturally seen as an essential motivation for approaches aiming at building a broader, more generic repository from existing, more specialized repositories. However, it may also apply to the inverse approaches aimed at splitting an ontology into smaller modules. In this case, the decomposition criterion should be based on the expected reusability of a module, i.e. how well can a module fill the purpose of various applications. Reusability emphasizes the need for rich mechanisms to describe a module in a way that maximizes the chances for modules to be understood, selected and used by other services and applications.

9.2.4 Understandability

An obvious prerequisite is the ability to use ontology to understand the concepts and properties it conveys. Whether the content is shown in visual or textual format, understanding is easier if the ontology is small (a module). Small ontologies are undoubtedly preferable if the user is a human being. However, size is not the only criterion that influences understandability. The way it is structured contributes to improving or decreasing understandability.

9.3 Style Ontology

The Style Ontology defines concepts of Style, Portrayal Rule Set, Portrayal Rule, Rule Condition and PortrayalContext. Figure 8 shows an overview of the model in UML.

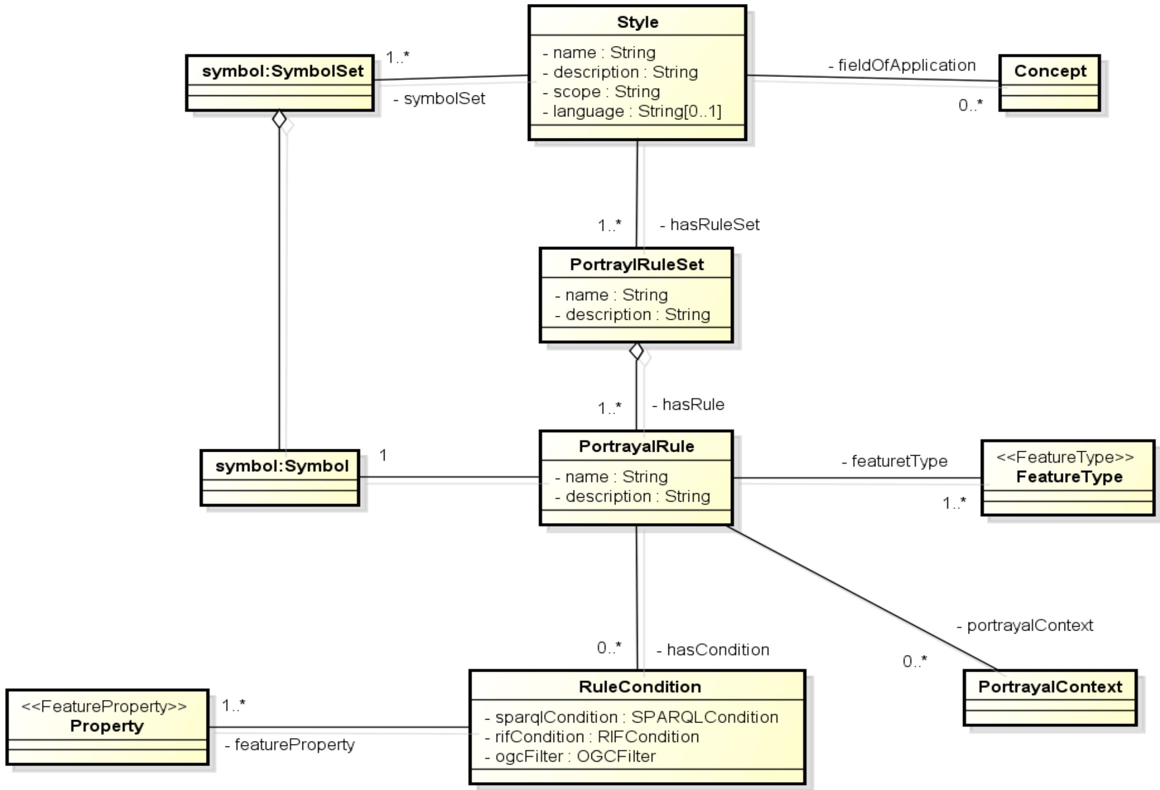


Figure 8 Style Model Overview

9.3.1 Style

The **Style** concept is central to the Style ontology. **Style** associates symbol sets with portrayal rule sets, which define the mapping of feature types to symbols. **Style** also captures descriptive metadata and tradecraft information such as the audience for the style, scope of application and field of application. Table 2 summarizes the properties of the **Style** class.

Table 2: Style properties

Name	**Definition**	**Type**	**Multiplicity**
name	Name identifier of the style.	String	One
dct:title	Multilingual human-readable title for the style		1..n (one per language)
dct:description	Multilingual human-readable description for the style	String	1..n (one per language)
dct:audience	The intended audience of this style.	foaf:Group	0..n

scope	Descriptive definition of the scope of application of the style	String	0..1
language	Language associated with the style	String	0..n
style:symbolSet	SymbolSet associated with the style	SymbolSet	1..n
fieldOfApplication	The field of application of this style, where values are defined as SKOS concept in a taxonomy.	skos:Concept	0..n
hasRuleSet	PortrayalRuleSet instances associated with the Style.	PortrayalRule Set	1..n

The following example shows the definition of the EMS Style with audience information organized as an hierarchy.

```
@prefix :      <http://www.opengis.net/testbed/11/cci/ems/style#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix style: <http://www.opengis.net/ont/portrayal/style#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix feature: <http://www.opengis.net/ont/feature#> .
@prefix group: <http://www.socialml.org/ontologies/group#> .
@prefix ems:   <http://www.opengis.net/testbed11/ont/incident/ems#> .

:EMSStyle  a             style:Style ;
        dct:audience
<http://ows.usersmarts.com/ldapp/audiences/community/CanadianEmergencyAndDisast
erManagement> ;
        dct:description   "Style defining the set of rules for mapping incident types from
EMS to symbology" ;
        dct:title         "EMS Style" ;
        style:hasRuleSet  :EMSRuleSet ;
        style:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet> .


<http://ows.usersmarts.com/ldapp/audiences/community/EmergencyAndDisasterManage
ment>
        a           foaf:Group , group:Group ;
        rdfs:label  "Emergency and Disaster Management Community" ;
        foaf:name   "Emergency and Disaster Management Community" .
```



```
<http://ows.usersmarts.com/ldapp/audiences/community/CanadianEmergencyAndDisasterManagement>
        a               foaf:Group , group:Group ;
        rdfs:comment    "Canadian Emergency and Disaster Management Community" ;
        rdfs:label      "Canadian Emergency and Disaster Management Community" ;
        group:subGroupOf
<http://ows.usersmarts.com/ldapp/audiences/community/EmergencyAndDisasterManagement> ;
        foaf:name       "Emergency and Disaster Management Community" .
```

9.3.2 PortrayalRuleSet

A **portrayal rule set** describes a function set which maps the feature types of a feature catalog to a symbol. A PortrayalRuleSet is composed of one or more portrayal rules, which in turn maps an individual feature type to a symbol. Table 3 provides a summary of the **PortrayalRuleSet**

Table 3 PortrayalRuleSet properties

Name	Definition	Type	Multiplicity
dct:title	Multilingual human readable name for the PortrayalRuleSet.	string	0..n (one per language)
dct:description	Multilingual human readable description for the PortrayalRuleSet.	string	0..n (one per language)
hasRule	PortrayalRule member of this PortrayalRuleSet.	PortrayalRule	0..n

The following is a sample of the PortrayalRuleSet defined for the EMS Style

```
@prefix :      <http://www.opengis.net/testbed/11/cci/ems/style#> .
@prefix style: <http://www.opengis.net/ont/portrayal/style#> .
@prefix dct:   <http://purl.org/dc/terms/> .

:EMSRuleSet  a          style:PortrayalRuleSet ;
        dct:description  "Set of rules for mapping incident types from EMS to symbology" ;
        dct:title       "EMS Portrayal Rule Set" ;
        style:hasRule   :ems.incident.temperature.windChill-portrayal-rule ,
                        :ems.incident.roadway.hazardousRoadConditions-portrayal-rule ,
                        :ems.incident.civil-portrayal-rule ,
                        :ems.incident.roadway.trafficReport-portrayal-rule ,
                        :ems.incident.meteorological.waterspout-portrayal-rule ,
```


…
:ems.incident.geophysical.lahar-portrayal-rule ,

:ems.incident.meteorological-portrayal-rule ,
:ems.incident.meteorological.stormSurge-portrayal-rule ,
:ems.incident.aviation-portrayal-rule ,
:ems.incident.hazardousMaterial.radiologicalHazard-portrayal-rule ,
:ems.incident.crime.bomb-portrayal-rule .

9.3.3 PortrayalRule

A **PortrayalRule** defines a rule that associates a feature type (**feature:FeatureType**) to a symbol *(***symbol:Symbol***)* satisfying a certain condition (**PortrayalRuleCondition**) in a given context (**PortrayalContext**). Table 4 summarizes its properties.

Table 4 PortrayalRule properties

Name	**Definition**	**Type**	**Multiplicity**
dct:title	Multilingual human readable name for the PortrayalRule.	string	0..n (one per language)
dct:description	Multilingual human readable description for the PortrayalRule.	string	0..n (one per language)
featureType	The featureType associated with this PortrayalRule	feature:FeatureType	0..n
portrayalContext	The context of application of the PortrayalRule	PortrayalContext	0..n
hasCondition	The conditions that needs to be satisfied by the rule	PortrayalRuleCondition	0..n
symbol	The symbol associated with the rule	symbol:Symbol	1

The listing below shows an example of **PortrayalRule** for Windchill. The PortrayalRule applied on the ems:EMSIncident featureType and associates the EMS symbol for Windchill defined by the URL: *http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.temperature.windChill-symbol*

:ems.incident.temperature.windChill-portrayal-rule
a ***style:PortrayalRule ;***
dct:description "Portrayal rule for incident type ems.incident.temperature.windChill" ;
dct:title "Wind Chill incident portrayal rule" ;
style:featureType ems:EMSIncident ;
style:hasRuleCondition :ems.incident.temperature.windChill-portrayal-rule-condition;
style:symbol
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.temperature.windChill -symbol> .

9.3.4 PortrayalRuleCondition

The **PortrayalRuleCondition** defines the condition in which a portrayal rule applies for a given feature. The **PortrayalRuleCondition** can be encoded using multiple encodings. Table 5 summarizes the properties of **PortrayalRuleCondition**

Table 5 PortrayalRuleCondition Properties

Name	**Definition**	**Type**	**Multiplicity**
featureProperty	The feature property affected by this condition	FeatureProperty	0..n
hasSPARQLCondition	SPARQL Encoding of the PortrayalRuleCondition	SPARQLCondition	0..1
hasRIFCondition	RIF Encoding of the PortrayalRuleCondition	RIFCondition	0..1
hasOGCFilterCondition	OGC Filter encoding of the PortrayalRuleCondition	OGCFilter	0..1

The following example demonstrates the encoding of the portrayal rule condition for the Portrayal rule for the symbol Windchill. The condition applies on the feature property ems:incidentType. If the value of this property is equals to http://www.opengis.net/taxonomy/ems#ems.incident.temperature.windChill then the rule is applicable.

```
:ems.incident.temperature.windChill-portrayal-rule-condition
      a   style:PortrayalRuleCondition ;
      style:featureProperty      ems:incidentType ;
      style:hasOGCFilterCondition  [ a        style:OGCFilter ;
                              style:body
"<ogc:Filter><ogc:PropertyIsEqualTo><ogc:PropertyName>incidentType</ogc:Prop
ertyName><ogc:Literal>http://www.opengis.net/taxonomy/ems#ems.incident.temperatur
e.windChill</ogc:Literal></ogc:PropertyIsEqualTo></ogc:Filter>"
                                             ] ;
       style:hasRIFCondition       [ a        style:RIFCondition ;
                              style:body  "Prefix(ems
<http://www.opengis.net/testbed11/ont/incident/ems#>)\nExists ?this (
ems:incidentType(?this
<http://www.opengis.net/taxonomy/ems#ems.incident.temperature.windChill>) )"
                                             ] ;
       style:hasSPARQLCondition     [ a        style:SPARQLCondition ;
                              style:body  "PREFIX ems:
<http://www.opengis.net/testbed11/ont/incident/ems#>\nASK \nWHERE { ?this
ems:incidentType
<http://www.opengis.net/taxonomy/ems#ems.incident.temperature.windChill>.\n}"
              ]
    ] ;
```

The rule is expressed in three different encodings: OGC Filter, SPARQL and RIF.

The SPARQL query is formulated as a ASK query which returns a boolean value. The variable ?this is bound to the current instance of featureType that is being tested.

```
PREFIX ems: <http://www.opengis.net/testbed11/ont/incident/ems#>
ASK
WHERE {
    ?this ems:incidentType
          <http://www.opengis.net/taxonomy/ems#ems.incident.temperature.windChill>.
}
```

The equivalent RIF condition is expressed as:

```
Prefix(ems <http://www.opengis.net/testbed11/ont/incident/ems#>)
Exists ?this
 ( ems:incidentType(?this
<http://www.opengis.net/taxonomy/ems#ems.incident.temperature.windChill>) )
```


The **SPARQLCondition** and **RIFCondition** can be used by a semantic portrayal rule engine that consumes feature data represented as Linked Data (recommendation for next testbed). We foresee that the portrayal catalog containing these rules can be extended with a rendering service that will apply the rules on a set of linked data compatible with the style and generates the portrayal rendering to a number of target formats (SVG, KML etc..). We suggest we experiment this capability in the next Testbed.

The **PortrayalRuleCondition** was used by a Web Processing Service (WPS) to generate the SLD document for a given Style, by using the **OGCFilter** associated with the Rule. To perform the bridge between the Linked Data representation of the FeatureType and GML representation we annotated the FeatureType and FeatureProperty with the attribute gmlName to indicate what is the mapping between the conceptual definition and the GML syntactic definition based on XML schema.

The following example shows the feature type definition for EMSIncident with the gmlName annotations.

```
ems:EMSIncident  a       feature:FeatureType ;
      rdfs:comment     "Incident defined for Canadian Emergency Management System" ;
      rdfs:label       "EMSIncident" ;
      feature:gmlName  "ems:EMSIncident" .

ems:incidentType  a      feature:FeatureProperty ;
      rdfs:label       "incidentType" ;
      feature:gmlName  "ems:incidentType" .
```

The following example shows the feature type definition for HSWGIncident with the gmlName annotations.

```
hswg:HSWGIncident  a     feature:FeatureType ;
      rdfs:comment     "Incident defined for Homeland Security Working Group" ;
      rdfs:label       "HSWGIncident" ;
      feature:gmlName  "hswg:HSWGIncident" .

hswg:incidentType  a     feature:FeatureProperty ;
      rdfs:label       "incidentType" ;
      feature:gmlName  "hswg:incidentType" .
```

9.4 Symbology Ontology

The Symbology Ontology is a microtheory that defines the conceptual model for defining **SymbolSet** and **Symbol** with their structural components called **SymbolComponent**. Figure 9 shows an overview of the model.

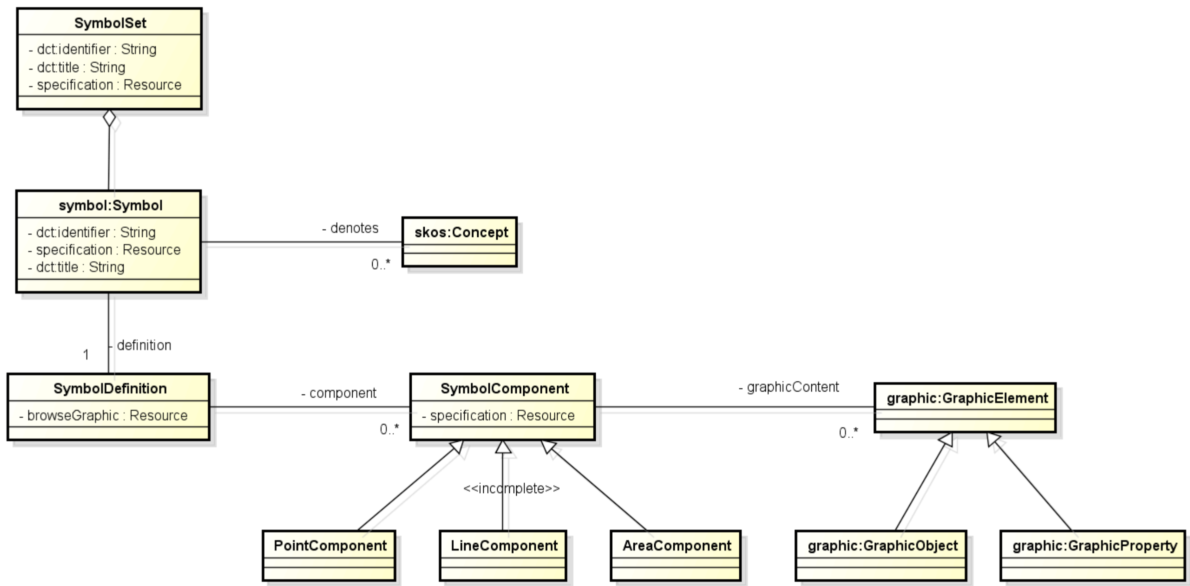


Figure 9 Symbology Model Overview

9.4.1 SymbolSet

SymbolSet collects symbols into sets of symbols that are used together. Symbols can be shared among symbol sets. A Symbol set can be equated with legend of a map.Table 6 summarizes the **SymbolSet** properties.

Table 6 SymbolSet Properties

Name	**Definition**	**Type**	**Multiplicity**
dct:identifier	Unique identifier for the symbol set mainly used by machine.	string	One
dct:title	Multilingual human readable name for the PortrayalRule.	string	0..n (one per language)
dct:description	Multilingual human readable description for the PortrayalRule.	string	0..n (one per language)

specification	Cites the specification standard for the SymbolSet	Resource	0..1
hasSymbol	Symbol member of this SymbolSet	Symbol	0..n

The following example shows a sample of the EMS SymbolSet.

```
@prefix :      <http://www.opengis.net/testbed/11/cci/ems/symbols#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix graphic: <http://www.opengis.net/ont/portrayal/graphic#> .
@prefix symbol: <http://www.opengis.net/ont/portrayal/symbol#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .

:EMSSymbolSet  a            symbol:SymbolSet ;
       dct:description      "Standard Canadian Emergency Mapping Symbology (EMS)
SymbolSet version 1.0" ;
       dct:title            "Canadian Emergency Mapping Symbology (EMS) SymbolSet
(version 1.0)" ;
       symbol:hasSymbol      :ems.incident.roadway.roadwayClosure-symbol ,
:ems.incident.temperature.windChill-symbol , :ems.incident.temperature.heatWave-
symbol , :ems.incident.civil.looting-symbol , :ems.incident.civil.dignitaryVisit-symbol ,
:ems.incident.civil.displacedPopulations-symbol , :ems.incident.publicService-symbol ;
   symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> .
```

9.4.2 Symbol

A **Symbol** is the type used to define symbol classes. Symbols are collected into symbol sets. A symbol has one machine readable identifier. A Symbol is described by a title, description and can refer to a formal specification document. A symbol can denote a concept defined in a SKOS taxonomy. Table 7 summarizes the Symbol properties.

Table 7 Symbol Properties

Name	**Definition**	**Type**	**Multiplicity**
dct:identifier	Machine readable name for the symbol. The identifier should be unique	string	1
dct:title	Multilingual human readable	string	0..n (one per

	title for the symbol		language)
dct:description	Multilingual human readable description for the symbol	string	0..n (one per language)
specification	Reference to the full details of the portrayal symbol	Resource	0..1
denotes	Concept that is denoted by the symbol	skos:Concept	0..n
definition	Symbol Definition defining the composition structure of the symbol	SymbolDefinition	0..1
symbolSet	The SymbolSet which this symbol belongs to.	SymbolSet	0..n
skos:notation	Notation used to refer the symbol as defined in a notation system. Use a custom datatype if multiple notations are used	string	0..n

The following listing shows the encoding in Turtle for the WindChill symbol belonging to the EMS SymbolSet.

```
:ems.incident.temperature.windChill-symbol
	a                   symbol:Symbol ;
	rdfs:label           "windChill" ;
	dct:identifier       "ems.incident.temperature.windChill" ;
	symbol:definition     :ems.incident.temperature.windChill-symbolDefinition ;
	symbol:denotes
<http://www.opengis.net/taxonomy/ems#ems.incident.temperature.windChill> ;
	symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> ;
	symbol:symbolSet      :EMSSymbolSet ;
	skos:notation         "ems.incident.temperature.windChill"^^:emsNotation .
```

9.4.3 SymbolDefinition

SymbolDefinition is root type for types that defines the composition of symbols. It represents the technical definition of a symbol. A symbol definition is composed as collection of symbol components, which contain the graphic elements and attributes used to define a symbol. This concept has been aligned with the ISO 19117 standard. Table 8 summarizes the properties of the **SymbolDefinition**.

Table 8 SymbolDefinition Properties

Name	Definition	Type	Multiplicity
dct:description	Multilingual human readable description of the symbol	string	0..n (one per language)
browseGraphic	Specifies graphics that may be used as metadata for the symbol and used to give a sample of the appearance of the symbol. The URL of the resource should be resolvable.	RDF Resource	0..n
component	Refers to the graphic component which makes up the symbol definition. A symbol definition with no components portrays nothing for a given feature.	SymbolComponent	0..n

The following sample encoded in Turtle format shows the symbol definition of the windChill symbol, which is composed of a pointIcon .

:ems.incident.temperature.windChill-symbolDefinition
a ***symbol:PointSymbolDefinition*** *;*
dct:description "Technical definition for symbol ems.incident.temperature.windChill";
symbol:browseGraphic
<http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.temperature.windChill.png> ;
symbol:component :ems.incident.temperature.windChill-pointIcon .

9.4.4 SymbolComponent

A **SymbolComponent** is the root type for types that defined the graphic representation of symbols. A Symbol Component is comprised of graphic elements, which are graphic objects and attributes used to define a symbol component. A symbol component can also refer to a formal specification document. Table 9 summarizes the properties of the SymbolComponent.

Table 9 SymbolComponent properties

Name	Definition	Type	Multiplicity
specification	Cites the specification standard for the graphics definition language used to define the symbol component.	string	0..1
graphicContent	Specified the graphic elements which make up the symbol component	graphic:GraphicElement	0..n

In this Testbed, we only focused on the implementation of icon-based symbol component. We defined a subclass of SymbolComponent called **PointIcon** to represent icon-based symbol component. It has a property graphic content referring to an icon image.

The following example defines the Point Icon for WindChill symbol in EMS.

:ems.incident.temperature.windChill-pointIcon
a symbol:PointIcon ;
symbol:graphicContent
<http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.temperature.windChill.png> ;
symbol:specification <https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> .

9.5 Graphics Ontology

The Graphics Ontology defines a vocabulary to describe graphic elements at the semantic level. Due to the limited time for this Testbed, the Graphics ontology focused mainly on defining two concepts: **ExternalGraphic** (for EMS raster icons) and **Font** (for HSWG fonts). The graphics Ontology needs to be extended in future Testbed to represent graphic objects and attributes for 0D, 1D and 2D such as line thickness, color, etc… As we develop these future extensions, we may consider breaking down the graphic ontology into several microtheories in order to favor reusability and scalability.

9.5.1 External Graphic

To support the rendering of EMS symbols provided in PNG format, we introduce the concept of **ExternalGraphic,** which represents a graphic that can be accessed online. The ExternalGraphic has a property onlineResource to indicate the URL of the resource and a property format to indicate the mime type of the resource. Multiple format of the same resource can be fetched, thus we allow the cardinality of this property to be 1 to n. Table 10 summarizes the property of **ExternalGraphic** concept.

Table 10 ExternalGraphic properties

Name	Definition	Type	Multiplicity
format	The format of the external graphic expressed as MIME type	string	1..n
onlineResource	The url to access the resource. The URL should be resolvable.	Resource	1

The following example shows how the RoadWayClosure symbol graphic is represented using ExternalGraphic.

<http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.roadway.roadwayClosure.png>
a ***graphic:ExternalGraphic*** *;*
rdfs:label "ems.incident.roadway.roadwayClosure icon" ;
dct:description "icon for ems.incident.roadway.roadwayClosure" ;
graphic:format "image/png" ;
graphic:onlineResource
<http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.roadway.roadwayClosure.png> .

9.5.2 Font, FontFamily and Foundry

For this testbed, we refactor the **Font**, **FontFamily** and **Foundry** ontology from the Testbed 10 into the graphic ontology. The **Font** and **FontFamilty** properties are summarized in Table 11and Table 12.

Table 11 Font properties

Name	Definition	Type	Multiplicity
fontCode	The code of the font within the font family.	string	1
fontFamily	The family of the Font.	FontFamily	1

Table 12 FontFamily Properties

Name	Definition	Type	Multiplicity
familyName	The family name	string	1
foundry	The font foundry that created the font family.	Foundry	0..1

The following example shows the **SymbolDefinition** for the HSWG shooting symbol associated with a **Font** defined in True type family.

```
:ShootingSymbolDefinition
    a               symbol:PointSymbolDefinition ;
    dct:description   "Technical definition for symbol ShootingSymbol" ;
    symbol:component  :ShootingSymbol-pointText .

:ShootingSymbol-pointText
    a                   symbol:PointText ;
    symbol:graphicContent  <tty:ERS_v2_Incidents#0x4B> ;
    symbol:specification   <http://www.fgdc.gov/HSWG/ref_pages/Incidents_ref.htm> .

<tty:ERS_v2_Incidents#0x4B>
    a               graphic:Font ;
    rdfs:label         "ShootingSymbol font" ;
    graphic:fontCode    "0x4B" ;
    graphic:fontFamily  <tty:ERS_v2_Incidents> .

<tty:ERS_v2_Incidents>
    a               graphic:FontFamily ;
    graphic:familyName  "ERS v2 Incidents" ;
    graphic:foundry     <http://symbolstore.org> .


<http://symbolstore.org>
    a          graphic:Foundry ;
    rdfs:label  "symbolstore.org" .
```

9.6 Portrayal Catalog Ontology

The Portrayal Catalog ontology provides the core facilities to define portrayal catalogs that contain elements necessary for a portrayal. The catalogue contains references to

Styles, Portrayal Rules, Symbol Sets with Symbols and Symbol definitions. The aim of this ontology is to provide support for discovery of styles, symbols and symbol sets for supporting portrayal. The fact that this information is encoded semantically enables the support of reasoning and extensions such as tradecraft information (audience, purpose, functions, qualities information). Due to the lack of time, this ontology was not fully formalized. This will need to be addressed in a future Testbed.

10 Portrayal Encoding

10.1 HSWG Portrayal Encoding

The HSWG Portrayal encoding was derived programmatically from the SKOS encoding of the taxonomy of HSWG Emergency Symbology (which was captured manually the testbed 10), as the symbols and incident types are matching one to one. The final results were split into three files:

- The SKOS encoding of the HSWG Incident taxonomy;
- The Portrayal rules for HSWG which defines the HSWG Style and Portrayal Rules; and
- The HSWG Symbol Set which defines the HSWG Symbol Set, associated symbols, Symbol definitions, components and graphics.

These three files were uploaded on the server in a RDF store and exposed through a SPARQL endpoint and Portrayal catalog REST API.

10.2 EMS Portrayal Encoding

The Canadian EMS Portrayal encoding was derived programmatically from the SKOS encoding of the taxonomy of EMS (which was captured manually for this testbed), as the symbols and incident types are matching one to one. The final results were split into three files:

- The SKOS encoding of the Canadian EMS Incident taxonomy;
- The Portrayal rules for EMS which defines the EMS Style and Portrayal Rules; and
- The EMS Symbol Set which defines the EMS Symbol Set, associated symbols, Symbol definitions, components and graphics.

These three files were uploaded on the server in a RDF store and exposed through a SPARQL endpoint and Portrayal catalog REST API.

11 Semantic Mediation

11.1 Introduction

Semantic Mediation was addressed to some extent in OWS-8, OWS-9 and OWS-10 (see section 2 for references). These Testbeds were mostly focused on performing semantic mediation for taxonomies. For example, gazetteers (such as GNIS, GNS, Geonames) often use different taxonomies for classifying feature types. To support semantic mediation, mappings are required from one concept in a source taxonomy to another one in the target taxonomy (using SKOS mapping relationships such as skos:exactMatch, skos:broadMatch, skos:closeMatch). The semantic mediation has been demonstrated using Web Feature Service-Gazetteer (WFS-G), however the mediation is performed as black box on syntactic representation of the features (using GML). Some extensions to the OGC Filter standard were added to accommodate the mediation of taxonomies. In OWS-10, the hydrology sub-thread of CCI attempted to address a more general approach for mediation by defining some mapping between two different hydrologic models. However, the solution was based on some UML tools that perform the mapping and no formal model was defined to encode the semantic mapping.

To address the semantic mediation of symbology in this testbed, we decided to address semantic mediation for linked data representation of information. We define semantic mediation as the transformation from one conceptual model to another, in particular from one ontology to another. Instances of the target classes are created from the values of instances of the source classes. We also wanted to formally addressthe semantic mediation for taxonomies by providing extensions to SPARQL to perform the semantic mapping.

One of the goals of the semantic mediation approach is to provide an extensible, sharable encoding of the semantic mappings that can be processed by machine. For this purpose, we leverage the existing linked data standards (RDF, OWL, SPARQL) to represent semantic mappings. These semantic mappings can be managed by a semantic mediation service to perform transformation between two models. For this testbed, we demonstrated a RESTful Semantic Mediation Service that performed semantic mapping between the HSWG Incident Model to the EMS Incident Model.

11.2 Review of existing approaches

This section provides an overview of the current existing approaches that attempt to address semantic mediation. A similar review was done in the OWS-8 ER. More up-to-date information is provided here.

11.2.1 EDOAL

The Expressive and Declarative Ontology Alignment Language (EDOAL) (http://alignapi.gforge.inria.fr/edoal.html) allows for representing correspondences between the entities of different ontologies. Unlike other formats, the alignment vocabulary allows the representation of complex correspondences allowing precise description of the relation between entities. The alignment vocabulary extends the alignment format.

Representing ontology alignments is the general purpose of this vocabulary. Particularly, it extends the ontology alignment format in order to enable the representation of complex correspondences.

This format can be used for cases where expressing equivalence or subsumption between terms is not sufficient and more precise relations need to be expressed. While term equivalence or subsumption might be enough for exchanging documents, more precise relations are needed to exchange and integrate data.

This vocabulary was originally designed with the goal of representing patterns of correspondence between ontologies. Since then the vocabulary was both simplified and extended to obtain a minimal vocabulary on top of the alignment format, thus providing the ability to express all possible kinds of ontology alignments.

The alignment vocabulary has the following features:

- **Construction** of entities from other entities can be expressed through algebraic operators. Constructed entities allow overcoming the shallowness of some ontologies.

- **Restrictions** can be expressed on entities in order to narrow their scope. Narrowing the scope of an entity makes possible to better align this entity with the one corresponding in the other ontology.

- **Transformations** of property values can be specified. Property values using different encoding or units can be aligned using transformations. The current version of EDOAL only supports limited transformations. This will be improved soon.

- **Linkkeys** can be defined for expressing conditions under which, instances of the aligned entities should be considered equivalent.

In the alignment format, an alignment is a set of cells, each cell being a correspondence between two entities. The alignment vocabulary extends this scheme by allowing cells to contain compound entity descriptions. Each entity can be typed according to one of the following category: Class, Instance, Relation, Property. A relation corresponds to an object property in OWL, a property to a datatype property. Each entity can then be restricted, and transformation can be specified on property values.

The EDOAL format is currently only supported by the Java-based Alignment API provided by INRIA. The transformations allowed by EDOAL are limited and there is no mechanism to create new functions without modifying the source code of the Alignment API. For this reason, we didn't retain this approach for the semantic mediation.

11.2.2 Rule Interchange Format

The Rule Interchange Format (RIF) is a W3C Recommendation (2013). RIF is a family of rule languages, called dialects, with rigorously specify syntax and semantics. RIF dialects can be classified into two groups: logic-based dialects and dialects for rules with actions. The logic-based dialects include languages that employ some kind of logic, such as first-order logic (often restricted to Horn logic). The rules-with-actions dialects include production rule systems, such as Drools, as well as event-condition-action rules, such as Reaction RuleML.

Currently the only two logic dialects within RIF are the Basic Logic Dialect (RIF-BLD) and the RIF Core Dialect. The only rules-with-actions dialect defined within RIF is the Production Rule Dialect (RIF-PRD).

RIF dialects include:

- Basic Logic Dialect (BLD): BLD is designed to be simple. A BLD document consists of a number of rules. Each rule contains a set of Horn rules; existential qualification is supported, negation is not. RIF-BLD has a number of syntactical extensions to support features such as objects and frames. The main semantic extensions include datatypes and externally defined predicates.

- Production Rule Dialect (PRD): This is one of the major dialects of RIF, influenced by production rule technology that has been demonstrated by major software vendors. Production rules, as they are currently practiced in main-stream systems like Jess or JRules, are defined using ad-hoc computational mechanisms, which are not based on logic. For this reason, RIF-PRD is not part of the suite of logical RIF dialects and stands apart from them. However, significant effort has been extended to ensure interoperability with the other dialects where possible.

- Core Dialect (Core): This dialect is a subset of both RIF-BLD and RIF-PRD based on RIF-DTB 1.0, thus enabling limited rule exchange between logic rule dialects and production rules. RIF-Core corresponds to Horn logic without function symbols (often called 'Datalog') with a number of extensions to support features such as objects and frames as in F-logic, internationalized resource identifiers for concepts, and XML Schema datatypes.

- Datatypes and Built-Ins (DTB): This specification lists the datatypes, built-in functions and built-in predicates expected to be supported by RIF dialects such as the RIF-Core, RIF BLD and RIF PRD.

The following are examples of the use of RIF within the geospatial community.

- EC INSPIRE: The INSPIRE report on the state of the art in Schema Transformation services observed that there is no commonly-used interchange format for mapping definitions. The INSPIRE study on Schema Transformation Services recommended that RIF be adopted as the interchange format for mappings. The report identified RIF PRD (Production Rule Dialect) as the most suited to the combination of rule conditions and consequent actions. The INSPIRE report provides a RIF template rule that is applied to features imported from a GML document.

The RIF standard has been very slow to be adopted by industry and very few tools are available today to exploit RIF. One of the reasons is that RIF does not naturally leverage the existing infrastructure and linked data standards. For example, RIF does not come with an RDF representation out of the box as it uses mainly a compact alternate syntax. Thus you cannot leverage RDF knowledge base to store RDF rules along with the ontologies. In addition, the standard query language SPARQL is not used to represent transformation rules, preventing reusing functions extensions (such as in GeoSPARQL). This may change in the future, but right now the lack of tools makes it hard to choose RIF as standard way to perform semantic mediation.

11.2.3 SPIN

The SPIN Modeling Vocabulary is a light-weight collection of RDF properties and classes to support the use of SPARQL to specify rules and logical constraints. Based on an RDF representation of SPARQL queries, SPIN defines three class description properties: *spin:constraint* can be used to define conditions that all members of a class must fulfill. *spin:rule* can be used to specify inference rules using SPARQL CONSTRUCTs and DELETE/INSERTs. *spin:constructor* can be used to initialize new instances with default values. In addition to these class description properties, SPIN provides a powerful meta-modeling capability that can be used to build your own modeling language and SPARQL extensions. These meta-modeling features provide the ability to encapsulate reusable SPARQL queries into *templates*, and to derive new SPARQL functions as well as magic properties from other SPARQL queries and functions.

The SPIN vocabulary was submitted to the W3C as a Note in 2011 and was updated in November 2014. The SPIN vocabulary is close to of the Testbed requirement to leverage existing linked data standards (RDF representation and SPARQL). SPIN can persist rules with the data the rules apply to. However the SPIN RDF syntax for representing a query (breaking down every element of the query in a RDF concept) is the biggest barrier of adoption. This requirement was relieved in the latest edition. TopQuadrant has published an open source SPIN Engine for leveraging SPIN. However, to our knowledge there is no other implementation available of the SPIN Engine. Due to this fact, SPIN is largely seen as a proprietary vocabulary. Our proposed SPARQL Extension ontology can be seen as a cleanup of SPIN by removing its legacy constructs for representing query in RDF) and clarifying some of the terms that are used in SPIN.

11.2.4 Topbraid SPIN Map

SPINMap is a TopBraid technology that uses a visual programming approach for mapping one model to another using SPIN functions to perform model transformations. SPINMap provides visual tools to create semantic mapping using SPIN functions that could defined by the user interface (see Figure 10 and Figure 11). The tools demonstrate that our approach is viable for using declarative RDF rules based on SPARQL and building tools to create semantic mapping visually.

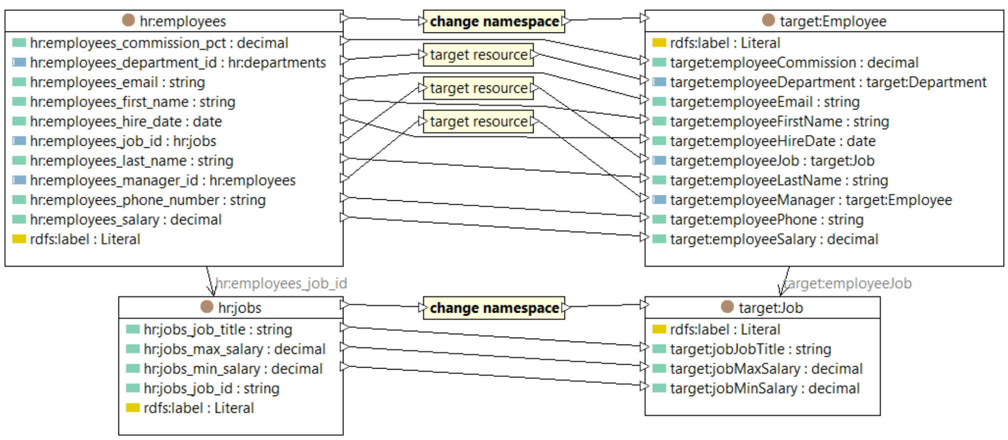


Figure 10: Use of SPIN Functions for Model transformations

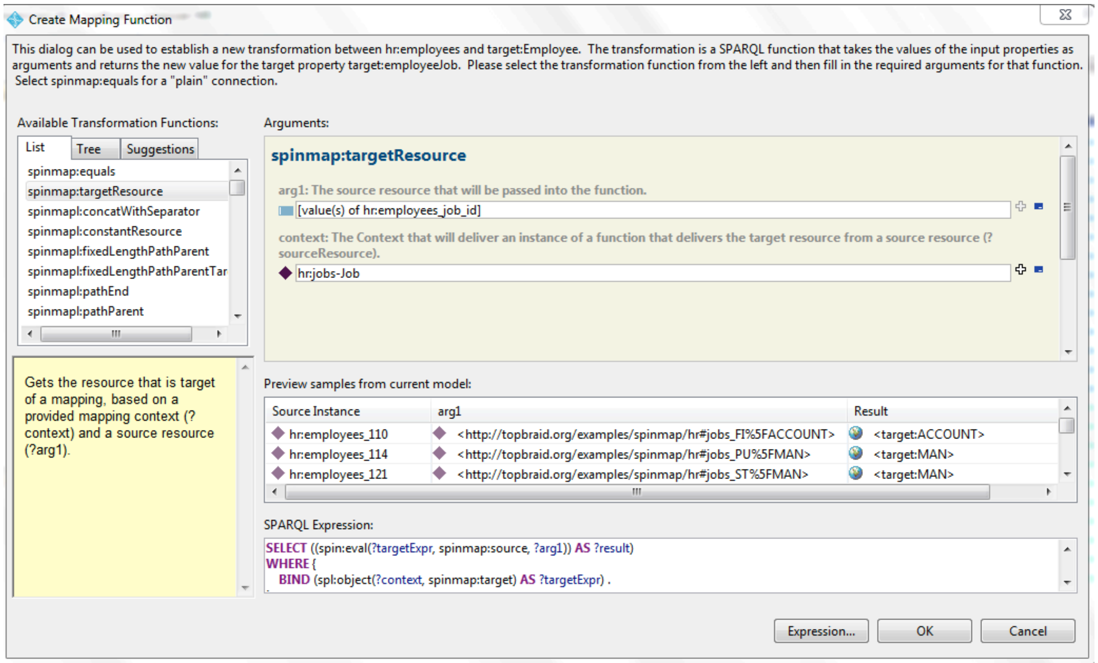


Figure 11 Topbraid SPINMap UI

11.3 Approach used for Testbed 11

The approach adopted to address semantic mediation is similar to the one used in SPIN. We want to define a SPARQL-based modeling vocabulary that is a lightweight collection of RDF properties and classes to support the use of SPARQL to specify functions, rules, logical constraints and mappings. We wanted instances of this vocabulary storable in existing RDF stores along with data or ontologies. By doing so, we can leverage Linked Data APIs standards and SPARQL to access the information without writing new services. We also wanted a mechanism able to describe domain specific constructs by using meta-modeling techniques allowing the introduction of higher level concepts that leverages lower-level concepts. To address this challenge, we introduce two new ontologies: the SPARQL Extensions Ontology and Semantic Mapping Ontology. Table 13 defines the namespace mapping for both microtheories.

Table 13 Namespace mapping for semantic mediation microtheories

Microtheory	Namespace	Prefix
SPARQL extension	http://www.opengis.net/ont/portrayal/style#	sparql-ext
mediation	http://www.opengis.net/ont/portrayal/symbol#	mediation

11.4 SPARQL Extensions ontology

The SPARQL Extensions ontology defines concepts that allow the definition of custom functions and mapping using SPARQL standards. This ontology provides the core building blocks for defining semantic mediation mappings but it has also other applications such as defining constraints on classes, annotating classes with inference rules, defining templates for pre-canned queries.

11.4.1 Modeling Query in RDF

One the core building blocks of the SPARQL Extension ontology is the ability to represent SPARQL query as Linked Data. Thus they can be linked and reused for different purposes, such as representing the body of a function, a rule or template.

The **Query** concept captures a SPARQL query as an RDF concept. Table 14 summarizes its properties:

Table 14 Query Properties

Name	Definition	Type	Multiplicity
textForm	The text form of the SPARQL query	string	1
queryLanguage	The query language specification used for encoding the text form of query	sd: Language	1

The SPIN vocabulary uses a different approach by breaking down every element of a SPARQL query into a separate concept (Filter, Function, Graph Pattern, Triple), making the ontology very hard to use. We see very little benefit taking this approach as it makes the implementation and usage of the SPIN ontology hard to use and less readable. We believe that capturing the SPARQL query in the text form along with the query language simplify considerably the editing and maintenance of the Query expressed in RDF. Note that the most recent version of the SPIN as added a text form property, acknowledging this problem, but still has the extra-baggage to representing every element of the query in the ontology. The SPARQL extensions ontology provides a clean version of representing Query without this extra baggage.

The sd:Language is defined by the W3C standard SPARQL Service Description. The following values are defined by the specification:

- **sd:SPARQL10Query** defines SPARQL 1.0 Query Language
- **sd:SPARQL11Query** defines SPARQL 1.1 Query Language

The Ontology introduces two subclasses to distinguish SPARQL10Query and SPARQL11Query

```
geosparql-ext:SPARQL10Query
        a                    owl:Class ;
        rdfs:label           "SPARQL10Query"^^xsd:string ;
        rdfs:subClassOf      geosparql-ext:Query ;
        owl:equivalentClass  [ a              owl:Restriction ;
                               owl:hasValue  sd:SPARQL10Query ;
                               owl:onProperty  geosparql-ext:queryLanguage
                             ] .
```

In addition to classification based on the query language used, the ontology defines three subclasses based on the type of query: **Ask**, **Construct** and **Select.** These subclasses are useful to constraint the type of query allowed for specific concept. A constraint may be tested with an Ask query. If the test fails, the Ask Query will return false. A Rule would only a Construct query, the body of the rule being the WHERE Clause and the CONSTRUCT pattern being the head of the rule.

11.4.2 Meta-modeling vocabulary

11.4.2.1 ParamerizableType

A ParamerizableType is a base metaclass that can have parameters. Examples are functions or query templates. A Parameterizable Type is a class that has predicate that acts as parameters (for example function, rules).

11.4.2.1.1 Parameter

The ontology defines the notion of **Parameter** that can be used by **ParameterizableType** such as **Function**, **Mapping**, or **Rule**. Each parameter has a unique name within the context of a **ParameterizableType**, a value type, a predicate that is used to associate the parameter value with the parameter name on instance of ParameterizableType (**FunctionCall** for example). The parameter may have a default value and may be optional. Table 15 summarizes the properties of **Parameter** class.

Table 15 Parameter properties

Name	Definition	Type	Multiplicity
name	The name of the parameter. The name needs to be unique within the ParameterizableObject.If not present, the local name of the predicate is used.	string	0..1
predicate	The associated predicate on the	rdf:Property	1

	instance of the ParameterizaleObject that binds the value of the parameter. The localName of the property is used a parameter name variable in the query.		
optional	Boolean indicating whether or not the parameter is optional. False by default.	xsd:boolean	0..1
valueType	The value type is defined by a Resource (which can be a Class or a Datatype).	Resource	1
defaultValue	The default value of the parameter if no value is bound to a parameter	Same type than the ValueType	0..1

11.4.3 Modeling Functions

The SPARQL language provides a mechanism to extend the query language with function extensions. For example, GeoSPARQL provides a set of spatial functions that extend the built-in SPARQL functions to support spatial queries. These functions are implemented based on proprietary mechanisms specific to SPARQL engines. For example, both Jena and Sesame APIs provide a Function Registry where you can programmatically register custom functions. The SPARQL Extension ontology provides a vocabulary to express functions in declarative manner that can be shared using Linked Data standards. The extensions can be automatically registered in SPARQL Engine using a wrapper mechanism. If a function encountered in a SPARQL engine is not known in a SPARQL engine, then the engine can follow the link to the function URI to get all the triples needed to understand how to execute it. This assumes that the URI of the function is de-referencable. The vocabulary enables the construction of a *web of functions* that can be leverage by SPARQL engines to augment their capabilities.

Functions play an important role in semantic mediation as many concept and property mappings require some transformations of one or more values to another one. The ability to express these functions declaratively favors reusability of mapping functions across mappings.

Best practice

Use resolvable URIs for identifying custom functions, so they can be fetched by SPARQL engine when they cannot be resolved locally.

11.4.3.1 Function

The **Function** concept is a **ParameterizableType** that can be used to define new SPARQL functions so that these new function can be used in expressions such as FILTER or BIND clauses. Functions are defined by a body and zero or more parameter descriptors (sometimes called arguments). The body of a function must be an Ask query, or a Select query with exactly one result variable that is compatible with the return type associated with the function. The function can also be associated with a string symbol.

The function themselves are classes that are instances of this metaclass. Function calls are instances of the function classes, with property values for the arguments. Table 16 summarizes the properties of Function.

Table 16 Function Properties

Name	Definition	Type	Multiplicity
body	The body of the function expressed as a SPARQL Select or Ask (predicate function)	Ask or Select	1
returnType	The value type of the return value of the function. If the body is a Ask instance, the returnType should xsd:boolean.	Resource	1
symbol	A symbol associated with function (used for display or parser).	string	0..1
hasParameter	Inherited Property from ParameterizableType that define the parameters of the function	Parameter	0..n

The following listing shows the definition of the function ChangeNamespace which takes two parameters: arg1 and targetNamespace.

```
:ChangeNamespace  a            sparql-ext:Function , owl:Class ;
      rdfs:comment           "Function changing the namespace of a uri" ;
      rdfs:label             "ChangeNamespace" ;
      rdfs:subClassOf        sparql-ext:FunctionCall ;
      sparql-ext:body        [ a                    sparql-ext:Query , sparql-ext:Select ;
                               sparql-ext:queryLanguage  sd:SPARQL11Query ;
                               sparql-ext:textForm
         "PREFIX fn:<http://www.usersmarts.com/ont/2005/06/ks/functor#>
          SELECT ?uri
          WHERE {
            BIND(URI(CONCAT(?targetNamespace,fn:localName(?arg1)) ) AS ?uri) .
           }"
       ] ;
      sparql-ext:hasParameter  [ a                 sparql-ext:Parameter ;
                               rdfs:comment        "URI Resource to change namespace" ;
                               rdfs:label          "arg1" ;
                               sparql-ext:predicate  sparql-ext:arg1 ;
                               sparql-ext:valueType  xsd:string
                             ] ;
      sparql-ext:hasParameter  [ a                 sparql-ext:Parameter ;
                               rdfs:comment        "Target namespace to substitute" ;
                               rdfs:label          "targetNamespace" ;
                               sparql-ext:predicate  :targetNamespace ;
                               sparql-ext:valueType  rdfs:Resource
                             ] ;
      sparql-ext:returnType    rdfs:Resource .
```

Once the function is defined, it can be used in SPARQL queries such as:

FILTER (BIND (ex:ChangeNamespace(?uri,'http://targetnamespace') AS ?newURI)

11.4.3.2 FunctionCall

The FunctionCall concept is used to describe an instance of a Function call, where parameters of the function are assigned to values through the predicate defined in the parameter. An instance of a Function (metaclass) is also a subclass of FunctionCall (see example in Function section).

11.4.3.3 FunctionLibrary

Functions can be organized into a library of functions. A **FunctionLibray** defines a set of functions that are used for a specific domain. It has descriptive metadata such as title and description. We may add in the future more tradecraft information such as audience, scope, domain information. Table 17 summarizes the **FunctionLibrary** properties.

Table 17 FunctionLibrary properties

Name	**Definition**	**Type**	**Multiplicity**
dct:title	Multilingual human readable title for the function library	string	0..n (one per language)
dct:description	Multilingual human readable description for the symbol	string	0..n (one per language)
hasFunction	Function member of this library	Function	1

11.4.4 Modeling Mappings

11.4.4.1 MappingType

MappingType is a **ParameterizableType** that enables the construction of custom reusable Mapping. An alignment between two ontologies is composed of a set of Mapping instances. Table 18 summarized the Mapping Type property.

Table 18 MappingType Properties

Name	**Definition**	**Type**	**Multiplicity**
name	Unique identifier of the mapping type	string	1
hasParameter	Inherited Property from ParameterizableType that define the parameters of the function	Parameter	0..n
mappingRule	SPARQL Rule encoding the mapping rules using the parameters defined with this mapping Type	SPARQLRule	1

The following example shows the definition of a ClassMapping **MappingType** instance. The ClassMapping takes three parameters:

- **sourceType** : the source type to map from (RDFS Class);

- **targetType**: the target type to map to (RDFS Class); and
- **expression**: the transformation function to use to modify the identifier of the instance (ChangeNamespace for example).

The Mapping Rule of the Class mapping is defined as a SPARQL Construct. The rule says that for every instance *(?this*) of class *?sourceType* (one of the parameter), create an instance *?target* with the class *?targetType* (one of the parameter). The target is defined as the result of the evaluation of the expression taking *?this* for its argument (if expression is defined), otherwise *?target* is the same than *?this*). We introduce a new function **eval** in GeoSPARQL to evaluate expression defined as function (defined with this ontology).

```
PREFIX fn:<http://www.knowledgesmarts.com/ontology/functions#>
PREFIX ksfun:   <http://www.usersmarts.com/ont/2005/06/ks/functor#>
PREFIX geosparqlFn:   <http://www.opengis.net/def/function/geosparql/>
CONSTRUCT {
?target a ?targetType.
}
WHERE {
?this a ?sourceType.
BIND
(ksfun:if(ksfun:bound(?expression),geosparqlFn:eval(?expression,fn:arg1,?this),?this)
AS ?target ).
}
```

```
:ClassMapping  a              sparql-ext:MappingType , owl:Class ;
      rdfs:comment            "Mapping transforming one instance of a class to another
class" ;
      rdfs:label              "ClassMapping" ;
      rdfs:subClassOf         sparql-ext:Mapping ;
      sparql-ext:hasParameter  [ a                sparql-ext:Parameter ;
                        rdfs:comment        "The source type to map from" ;
                        rdfs:label          "sourceType" ;
                        sparql-ext:predicate  :sourceType ;
                        sparql-ext:valueType  rdfs:Class
                      ] ;
      sparql-ext:hasParameter  [ a                sparql-ext:Parameter ;
                        rdfs:comment        "The target type to map to" ;
                        rdfs:label          "targetType" ;
                        sparql-ext:predicate  :targetType ;
                        sparql-ext:valueType  rdfs:Class
                      ] ;
      sparql-ext:hasParameter  [ a                sparql-ext:Parameter ;
```



```
                                rdfs:comment          "The transformation function to use to modify
the identifier of the instance" ;
                                rdfs:label          "expression" ;
                                sparql-ext:predicate  :expression ;
                                sparql-ext:valueType  sparql-ext:Function
                              ] ;
        sparql-ext:mappingRule   [ a              sparql-ext:SPARQLRule ;
                                sparql-ext:body  [ a sparql-ext:Construct ;
                                                 sparql-ext:queryLanguage  sd:SPARQL11Query ;
                                                 sparql-ext:textForm       "PREFIX
fn:\t\t<http://www.knowledgesmarts.com/ontology/functions#>\r\nPREFIX ksfun:
<http://www.usersmarts.com/ont/2005/06/ks/functor#> \r\nPREFIX geosparqlFn:
<http://www.opengis.net/def/function/geosparql/> \r\nCONSTRUCT {\r\n\t?target a
?targetType.\r\n}\r\nWHERE {\r\n\t?this a ?sourceType .\r\n\tBIND
(ksfun:if(ksfun:bound(?expression),geosparqlFn:eval(?expression,fn:arg1,?this),?this)
AS ?target ).\r\n}"
                                                 ]
                              ] ;
        sparql-ext:name          "ClassMapping" .
```

The following mapping type **PropertyMapping-1-1** maps a source predicate of a resource to a target predicate using optionally a transformation expression.

```
:PropertyMapping-1-1  a          sparql-ext:MappingType , owl:Class ;
        rdfs:comment            "Mapping a source predicate of a resource to a target
predicate using optionally a transformation expression" ;
        rdfs:label              "PropertyMapping11" ;
        rdfs:subClassOf         sparql-ext:Mapping ;
        sparql-ext:hasParameter  [ a                 sparql-ext:Parameter ;
                                rdfs:comment          "The source predicate to map from" ;
                                rdfs:label          "sourcePredicate" ;
                                sparql-ext:predicate  :sourcePredicate ;
                                sparql-ext:valueType  rdf:Property
                              ] ;
        sparql-ext:hasParameter  [ a                 sparql-ext:Parameter ;
                                rdfs:comment          "The target predicate to map to" ;
                                rdfs:label          "targetPredicate" ;
                                sparql-ext:predicate  :targetPredicate ;
                                sparql-ext:valueType  rdf:Property
                              ] ;
        sparql-ext:hasParameter  [ a                 sparql-ext:Parameter ;
                                rdfs:comment          "The transformation function to use to modify
the value of source predicate" ;
                                rdfs:label          "expression" ;
                                sparql-ext:predicate  :expression ;
                                sparql-ext:valueType  sparql-ext:Function
```



```
                    ] ;
    sparql-ext:hasParameter  [ a            sparql-ext:Parameter ;
                    rdfs:comment        "The mapping context in which this property
mapping applies" ;
                    rdfs:label          "context" ;
                    sparql-ext:predicate  :context ;
                    sparql-ext:valueType  :ClassMapping
                    ] ;
    sparql-ext:mappingRule   [ a          sparql-ext:SPARQLRule ;
                    sparql-ext:body  [ a              sparql-ext:Query , sparql-
ext:Construct ;
                              sparql-ext:queryLanguage  sd:SPARQL11Query ;
                              sparql-ext:textForm       "PREFIX fn:\t\t
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#>\r\nPREFIX
ksfun:   <http://www.usersmarts.com/ont/2005/06/ks/functor#> \t\r\nPREFIX mapping:
<http://www.opengis.net/testbed11/ont/geosparql/mapping/core#> \t\t\t\t\t\t\r\nPREFIX
geosparqlFn:   <http://www.opengis.net/def/function/geosparql/>
\r\n\r\n\r\nCONSTRUCT {\r\n\t?target ?targetPredicate ?newValue .\r\n}\r\nWHERE
{\r\n\t?this ?sourcePredicate ?oldValue .\r\n\t?this a ?sourceType .\r\n\tBIND
(fn:object(?context,mapping:sourceType) AS ?sourceType) .\r\n\t\t\t\t\t\t\t\r\n\tBIND
(ksfun:if(ksfun:bound(?expression),geosparqlFn:eval(?expression,fn:arg1,?oldValue),?ol
dValue) AS ?newValue) .\r\n\tBIND
(ksfun:if(ksfun:bound(fn:object(?context,mapping:expression)),geosparqlFn:eval(fn:obje
ct(?context,mapping:expression),fn:arg1,?this),?this) AS ?target) .\r\n}"
                              ]
                    ] ;
    sparql-ext:name         "PropertyMapping11" .
```

The following mapping type **PropertyMapping-0-1** creates a target predicate from an resource using optionally a transformation expression.

```
:PropertyMapping-0-1  a         sparql-ext:MappingType , owl:Class ;
    rdfs:comment            "Mapping creating a target predicate from an resource using
optionally a transformation expression" ;
    rdfs:label              "PropertyMapping01" ;
    rdfs:subClassOf         sparql-ext:Mapping ;
    sparql-ext:hasParameter  [ a            sparql-ext:Parameter ;
                    rdfs:comment        "The target predicate to map to" ;
                    rdfs:label          "targetPredicate" ;
                    sparql-ext:predicate  :targetPredicate ;
                    sparql-ext:valueType  rdf:Property
                    ] ;
    sparql-ext:hasParameter  [ a            sparql-ext:Parameter ;
```



```
                          rdfs:comment        "The transformation function to use to modify
the value of source predicate" ;
                          rdfs:label          "expression" ;
                          sparql-ext:predicate  :expression ;
                          sparql-ext:valueType  sparql-ext:Function
                        ] ;
    sparql-ext:hasParameter  [ a                  sparql-ext:Parameter ;
                          rdfs:comment        "The mapping context in which this property
mapping applies" ;
                          rdfs:label          "context" ;
                          sparql-ext:predicate  :context ;
                          sparql-ext:valueType  :ClassMapping
                        ] ;
    sparql-ext:mappingRule   [ a              sparql-ext:SPARQLRule ;
                          sparql-ext:body  [ a                  sparql-ext:Query , sparql-
ext:Construct ;
                                      sparql-ext:queryLanguage  sd:SPARQL11Query ;
                                      sparql-ext:textForm       "PREFIX fn:\t\t
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#>\r\nPREFIX
ksfun:   <http://www.usersmarts.com/ont/2005/06/ks/functor#> \t\t\r\nPREFIX mapping:
<http://www.opengis.net/testbed11/ont/geosparql/mapping/core#> \r\nPREFIX
geosparqlFn:   <http://www.opengis.net/def/function/geosparql/>
\r\n\t\t\r\nCONSTRUCT {\r\n\t?target ?targetPredicate ?newValue .\r\n}\r\nWHERE
{\r\n\t?this a ?sourceType .\r\n\tBIND (fn:object(?context,mapping:sourceType) AS
?sourceType) .\r\n\tBIND (geosparqlFn:eval(?expression,fn:arg1,?this) AS ?newValue)
.\r\n\tBIND (geosparqlFn:eval(fn:object(?context,mapping:expression),fn:arg1,?this) AS
?target) .\r\n}"
                                    ]
                        ] ;
    sparql-ext:name        "PropertyMapping01" .
```

11.4.4.2 Mapping

The Mapping class is used to flag instances of a **MappingType** instance. It is used mostly as a classifier to indicate that the instance of instance of MappingType is a Mapping. The MappingType instance should always be defined as a subclass of Mapping (see **ClassMapping** example above). The ontology provides two subclasses of Mapping: **PropertyMapping** to indicate mapping of properties and **ResourceMapping** to indicate mapping of resources.

11.4.5 Modeling Templates

A Template is used to represent a canned parameterized query.

11.4.5.1 QueryTemplate

The **QueryTemplate** is the base class for representing canned parameterized SPARQL query. Table 19 summarizes the property of **QueryTemplate**.

Table 19 Query Tempalte properties

Name	Definition	Type	Multiplicity
labelTemplate	A string template contains parameter variable that is aimed to be substituted for display the label associated with instance of Template.	String	0..1
templateQuery	The canned query that uses the parameters names as variables.	Query	1
hasParameter	Inherited Property from ParameterizableType that define the parameters of the function	Parameter	0..n

11.4.5.2 AskTemplate

An **AskTemplate** is a **QueryTemplate** that has a body using only Ask query returning a boolean. Table 20 summarizes the properties of **AskTemplate**.

Table 20 AskTemplate properties

Name	Definition	Type	Multiplicity
Body	The Ask body of the template using the parameter names as variables.	Ask	1
hasParameter	Inherited Property from ParameterizableType that define the parameters of the function	Parameter	0..n

11.4.5.3 ConstructTemplate

A **ConstructTemplate** is a **QueryTemplate** that has a body using only Construct query returning a graph. Table 21 summarizes the properties of **AskTemplate**.

Table 21 Construct Template properties

Name	Definition	Type	Multiplicity
Body	The SPARQL construct body of the template using the parameter names as variables.	Construct	1
hasParameter	Inherited Property from ParameterizableType that define the parameters of the function	Parameter	0..n

11.4.5.4 SelectTemplate

A **SelectTemplate** is a **QueryTemplate** that has a body using only Select query. Table 22 summarizes the properties of **SelectTemplate**.

Table 22 SelectTemplate properties

Name	Definition	Type	Multiplicity
Body	The SPARQL Select body of the template using the parameter names as variables.	Select	1
hasParameter	Inherited Property from ParameterizableType that define the parameters of the function	Parameter	0..n

11.4.6 Modeling Rules

A number of rule languages for Linked data have been proposed. The W3C Rule Interchange Framework (RIF) is a Recommendation. RIF provides several dialects to accommodate the different expressiveness of rule languages. It provides a default syntax that is not compatible with RDF. Unfortunately, the adoption of RIF has been very slow since its adoption as a recommendation and the maturity of the tools handling RIF format is very limited.

Our goal is to have a way to store rules using existing Linked Data infrastructure and well-adopted standard such as SPARQL/GeoSPARQL. The Topbraid SPIN has similar goal than our ontology however the model is crippled with complexity and concepts that are too close to the proprietary Topbraid software. We aimed at simplifying the ontology and clarifying the concepts used in the ontology.

One of the applications of using declarative rules is to annotate OWL classes with inference rules associated with a given class using the property **rule.**

11.4.6.1 Rule

The ontology defines the abstract class **Rule**. A *rule* is an IF - THEN construct. If some condition (the IF part) that is checkable in some dataset holds, then the conclusion (the THEN part) is processed. For this testbed, **SPARQLRule** is the only subclass of **Rule**. Other rule languages may be used in the future and the base class Rule provides mechanism to accommodate this new Rule Language by using subclassing mechanism.

11.4.6.2 SPARQL Rule

SPARQLRule is defined a subclass of Rule. It defines a Rule that can be expressed using SPARQL Construct. The body of the SPARQL query represents the body of the rule and the construct template represents the head of the rule. The use of SPARQL Construct for rules is very natural as it aligns with the Linked Data Model using standard SPARQL syntax. Table 23 defines the properties for **SPARQLRule**.

Table 23 SPARQLRule Properties

Name	Definition	Type	Multiplicity
body	The SPARQL Construct body associated with the rule	Construct	1

11.4.6.3 Rule Library

Rules can be organized into a library of rules. A **RuleLibrary** defines a set of rules that are used for a specific domain. It has descriptive metadata such as title and description. We may add in the future more tradecraft information such as audience, scope, domain information. Table 24 summarizes the **RuleLibrary** properties.

Table 24 RuleLibrary Properties

Name	Definition	Type	Multiplicity
dct:title	Multilingual human readable title for the symbol	string	0..n (one per language)
dct:description	Multilingual human readable description for the symbol	string	0..n (one per language)
hasRule	The text form of the SPARQL	Rule	1

	query		

11.5 Semantic Mediation Ontology

The Semantic Mediation Ontology defines the notion of **Alignment** between two ontologies. It leverages the SPARQL Extensions ontology by referring to Mapping instances. We may consider in the future migrating **Mapping** and **MappingType** concepts into this ontology as they are building block for semantic mapping.

11.5.1 Alignment

Name	**Definition**	**Type**	**Multiplicity**
dct:title	Multilingual human readable title for the symbol	string	0..n (one per language)
dct:description	Multilingual human readable description for the symbol	string	0..n (one per language)
alignmentName	The unique name of the alignment used by machine	Rule	1
sourceOntology	The source ontology for this alignment	owl:Ontology	1
targetOntology	The target ontology for this alignment	owl:Ontology	1
hasMapping	Mappings associated with the alignment	sparql-ext:Mapping	1..n

The following listing defines the complete alignment between HSWG and EMS Incident (as described in section 8.3).

```
:HSWG-EMS-Alignment a mediation:Alignment;
    mediation:alignmentName 'HSWG2EMS';
    rdfs:label 'Alignment between HSWG Incident and EMS Incident';
    rdfs:comment 'Alignment between HSWG Incident and EMS Incident';
    mediation:sourceOntology <http://www.opengis.net/testbed11/ont/incident/hswg#>;
    mediation:targetOntology <http://www.opengis.net/testbed11/ont/incident/ems#>;
    mediation:hasMapping  :HSWG-EMS-IncidentMapping,
```

```
                    :incidentId-mapping,
                    :date-mapping,
                    :time-mapping,
                    :label-title-mapping,
                    :description-mapping,
                    :incidentType-mapping,
                    :hasAddress-mapping,
                    :AddressMapping,
                    :city-mapping,
                    :fullAddress-mapping,
                    :state-mapping,
                    :hasPosition-mapping,
                    :PointMapping,
                    :latitude-mapping,
                    :longitude-mapping,
                    :asWKT-mapping.

:HSWG-EMS-IncidentMapping
     a              ogc-map:ClassMapping , sparql-ext:Mapping ;
     ogc-map:sourceType hswg:HSWGIncident ;
     ogc-map:targetType ems:EMSIncident ;
     ogc-map:expression
         [ a     fn:ChangeNamespace ;
          fn:targetNamespace "http://ows.usersmarts.com/testbed11/data/ems#"
         ] .
:date-mapping
    a      ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:incidentDate ;
    ogc-map:targetPredicate ems:incidentDate .

:time-mapping
    a      ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:incidentTime ;
    ogc-map:targetPredicate ems:incidentTime .

:label-title-mapping
    a      ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:title ;
    ogc-map:targetPredicate rdfs:label .

:incidentId-mapping
    a      ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
```



```
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:incidentNumber ;
    ogc-map:targetPredicate ems:incidentId .



:description-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:summary ;
    ogc-map:targetPredicate ems:description .

:incidentType-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:incidentType  ;
    ogc-map:targetPredicate ems:incidentType ;
    ogc-map:expression
          [ a      fn:skosMatch ;
            sparql-ext:arg2
<http://www.opengis.net/taxonomy/ems#EMSIncidentTaxonomy>
          ] .

:hasAddress-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:hasAddress ;
    ogc-map:targetPredicate ems:address .

:AddressMapping
      a                ogc-map:ClassMapping , sparql-ext:Mapping ;
      ogc-map:sourceType hswg:Address ;
      ogc-map:targetType vcard:Address.

:city-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :AddressMapping ;
    ogc-map:sourcePredicate hswg:city ;
    ogc-map:targetPredicate vcard:locality .

:fullAddress-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :AddressMapping ;
    ogc-map:sourcePredicate hswg:fullAddress ;
    ogc-map:targetPredicate vcard:street-address .
```

```
:state-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :AddressMapping ;
    ogc-map:sourcePredicate hswg:state ;
    ogc-map:targetPredicate vcard:region .

:hasPosition-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :HSWG-EMS-IncidentMapping ;
    ogc-map:sourcePredicate hswg:location ;
    ogc-map:targetPredicate ems:position .


:PointMapping
     a              ogc-map:ClassMapping , sparql-ext:Mapping ;
     ogc-map:sourceType geosparql:Point;
     ogc-map:targetType geosparql:Point.


:latitude-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :PointMapping ;
    ogc-map:sourcePredicate wgs84:lat ;
    ogc-map:targetPredicate wgs84:lat .

:longitude-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :PointMapping ;
    ogc-map:sourcePredicate wgs84:long ;
    ogc-map:targetPredicate wgs84:long .

:asWKT-mapping
    a       ogc-map:PropertyMapping-1-1 , sparql-ext:Mapping;
    ogc-map:context :PointMapping ;
    ogc-map:sourcePredicate geosparql:asWKT  ;
    ogc-map:targetPredicate geosparql:asWKT  .
```

11.6 Extensions functions to SPARQL

During this testbed, we implemented a number of function extensions to support the SPARQL extension vocabulary and the semantic mediation for SKOS taxonomy.

11.6.1 geosparql:skosMatch

To support semantic mediation for SKOS taxonomies, we define a new function **skosMatch** that could be leveraged by a SPARQL endpoint. The function is defined in the following namespace http://www.opengis.net/testbed/11/def/function/geosparql (extension of SPARQL for testbed 11). The function is defined as:

skosMatch(srcConceptURI, targetConceptSchemeURI)

and returns the best match for the source concept in the target concept scheme. The function first check if there is an **exactMatch**, if not check if there is **closeMatch**. If not check if there is a **broadMatch**, if not returns null. This function allows to do multiple tests into one single query, simplifying the client implementations.

11.6.1.1 Examples

Here some example of queries you can test under the following endpoint: http://ows.usersmarts.com/portrayal/api/sparql which provides a SPARQL client UI.

Example1: Find the matching concept from EMS *portClosure* in HSWG Taxonomy.

```
PREFIX symbol:<http://www.opengis.net/ont/portrayal/symbol#>
PREFIX style:<http://www.opengis.net/ont/portrayal/style#>
PREFIX geosparql-ext:<http://www.opengis.net/testbed/11/def/function/geosparql/>
PREFIX ems:<http://www.opengis.net/taxonomy/ems#>
PREFIX hswg:<http://www.fgdc.gov/HSWG/taxonomy/incidents#>
PREFIX geosparql: <http://www.opengis.net/ont/geosparql#>

DESCRIBE ?match
WHERE {
  BIND ( geosparql-ext:skosMatch( ems:ems.incident.marine.portClosure, hswg:Incident
) AS ?match)
}
```

The following result is returned:

```
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix symbol: <http://www.opengis.net/ont/portrayal/symbol#> .
@prefix style: <http://www.opengis.net/ont/portrayal/style#> .
@prefix hswg:  <http://www.fgdc.gov/HSWG/taxonomy/incidents#> .
@prefix dct:   <http://purl.org/dc/terms/> .
```


@prefix geosparql-ext: <http://www.opengis.net/testbed/11/def/function/geosparql/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix ems: <http://www.opengis.net/taxonomy/ems#> .

hswg:MarineIncident *a*
<http://www.opengis.net/ont/emergency/incident#IncidentTheme> ,
<http://www.w3.org/2002/07/owl#Thing> , skos:Concept ;
rdfs:label "Marine
incident"^^<http://www.w3.org/2001/XMLSchema#string> ;
skos:closeMatch ems:ems.incident.marine ;
skos:definition "An event involving a boat or ship resulting in damage, bodily
injury, death, or the disruption of transportation
service."^^<http://www.w3.org/2001/XMLSchema#string> ;
skos:exactMatch ems:ems.incident.marine ;
skos:inScheme hswg:Incident ;
skos:narrowMatch ems:ems.incident.marine.marineSecurity ,
ems:ems.incident.marine.portClosure *, ems:ems.incident.marine.specialMarine ;*
skos:narrower ems:ems.incident.marine.marineSecurity ,
ems:ems.incident.marine.portClosure , ems:ems.incident.marine.specialMarine ,
hswg:MarineAccident , hswg:MarineHijacking ;
skos:narrowerTransitive ems:ems.incident.marine.marineSecurity ,
ems:ems.incident.marine.portClosure , ems:ems.incident.marine.specialMarine ,
hswg:MarineAccident , hswg:MarineHijacking ;
skos:note "An event involving a boat or ship resulting in damage, bodily
injury, death, or the disruption of transportation
service."^^<http://www.w3.org/2001/XMLSchema#string> ;
skos:prefLabel "Marine
incident"^^<http://www.w3.org/2001/XMLSchema#string> ;
skos:semanticRelation hswg:MarineAccident , hswg:MarineHijacking ;
skos:topConceptOf hswg:Incident .

Example2: List all the matching of EMS concepts to HSWG taxonomy (callled hswg:Incident)

PREFIX symbol:<http://www.opengis.net/ont/portrayal/symbol#>
PREFIX style:<http://www.opengis.net/ont/portrayal/style#>
PREFIX geosparql-ext:<http://www.opengis.net/testbed/11/def/function/geosparql/>
PREFIX ems:<http://www.opengis.net/taxonomy/ems#>
PREFIX hswg:<http://www.fgdc.gov/HSWG/taxonomy/incidents#>
PREFIX geosparql: <http://www.opengis.net/ont/geosparql#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>


```
SELECT ?srcConcept ?match
WHERE {
 ?srcConcept a skos:Concept.
 ?srcConcept skos:inScheme ems:EMSIncidentTaxonomy.
 BIND ( geosparql-ext:skosMatch(?srcConcept, hswg:Incident ) AS ?match)
}
```

The following result is returned:


```
{
 "head": {
  "vars": [ "srcConcept" , "match" ]
 } ,
 "results": {
  "bindings": [
   {
    "srcConcept": { "type": "uri" , "value":
"http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilDemonstration" } ,
    "match": { "type": "uri" , "value":
"http://www.fgdc.gov/HSWG/taxonomy/incidents#CivilDemonstrations" }
   } ,
   {
    "srcConcept": { "type": "uri" , "value":
"http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilDisplacedPopulation" } ,
    "match": { "type": "uri" , "value":
"http://www.fgdc.gov/HSWG/taxonomy/incidents#CivilDisplacedPopulation" }
   } ,
   {
    "srcConcept": { "type": "uri" , "value":
"http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilEmergency" } ,
    "match": { "type": "uri" , "value":
"http://www.fgdc.gov/HSWG/taxonomy/incidents#CivilDisturbanceIncident" }
   } ,
   {
    "srcConcept": { "type": "uri" , "value":
"http://www.opengis.net/taxonomy/ems#ems.incident.civil.civilRioting" } ,
    "match": { "type": "uri" , "value":
"http://www.fgdc.gov/HSWG/taxonomy/incidents#CivilRioting" }
   } ,
   {
    "srcConcept": { "type": "uri" , "value":
"http://www.opengis.net/taxonomy/ems#ems.incident.crime.illigalImmigrant" } ,
    "match": { "type": "uri" , "value":
"http://www.fgdc.gov/HSWG/taxonomy/incidents#CriminalActivityIncident" }
```



```
    } ,
….. (truncated)
   ]
  }
}
```

This function comes handy to perform ontology alignment using mapping when some property uses different taxonomies. It makes it easier to write the transformation mapping using SPARQL.

11.6.2 geosparql:eval

SPARQL engines that supports SPARQL Extension Ontology must provide a built-in SPARQL function **geosparql:eval** that can be used to evaluate a expression or query at execution time. This makes it possible to define higher level functions that take other function calls and queries as arguments. For example, it is possible to place an expression as an argument into a template. The body of the template can then reference the expression (as a pre-bound variable) and evaluate it.

geosparql:eval takes any odd number of arguments. The first argument must be a reference to a expression (e.g., instance of a SPARQL function) or a **Select** or a **Ask**. The other arguments must come in pairs, so that the first one is a property and the second is a value that shall be pre-bound in the evaluation of the expression.

In the following example, an expression (bound to a variable ?expression) is executed, with the variable ?arg3 bound to the literal "value". The result of the function will be assigned to ?result.

```
BIND (geosparql:eval(?expression, sp:arg3, "value") AS ?result)
```

If the expression argument is a Select, then the result will be the first binding of the first result variable. If the expression argument is an Ask, then a typed boolean literal will be returned.

12 Implementations

12.1 Image Matters Semantic Mediation Service

The Semantic Mediation Service (SMS) is a new service introduced during this testbed. The service addresses the first task to perform the mediation of the information represented by the symbology. The SMS can be reused for different contexts when alignment from one ontology to another one is needed. For example, SMS can be used for search information expressed in one ontology to find information expressed in a different one. Future extensions may support SPARQL rewriting for a given alignment.

12.1.1 Architecture

We designed the SMS to be RESTful and to use Linked Data standards. The service is composed of two graph stores. The first one contains the definition of the alignments and mapping definitions. The second store contains the definition of the functions and mapping types. The mapping engine is used to perform the transformation from one Linked Data Model to another one. The mapping engine leverages the GeoSPARQL engine that is augmented with the plugins functions and rules. The service exposed a REST API to access the concepts from the knowledge stores using a Linked Data API. The service also provides a Mediation REST API that performs the mediation work for a specific alignment. Finally, the service provides a GeoSPARQL endpoint capable to query the Alignment database.

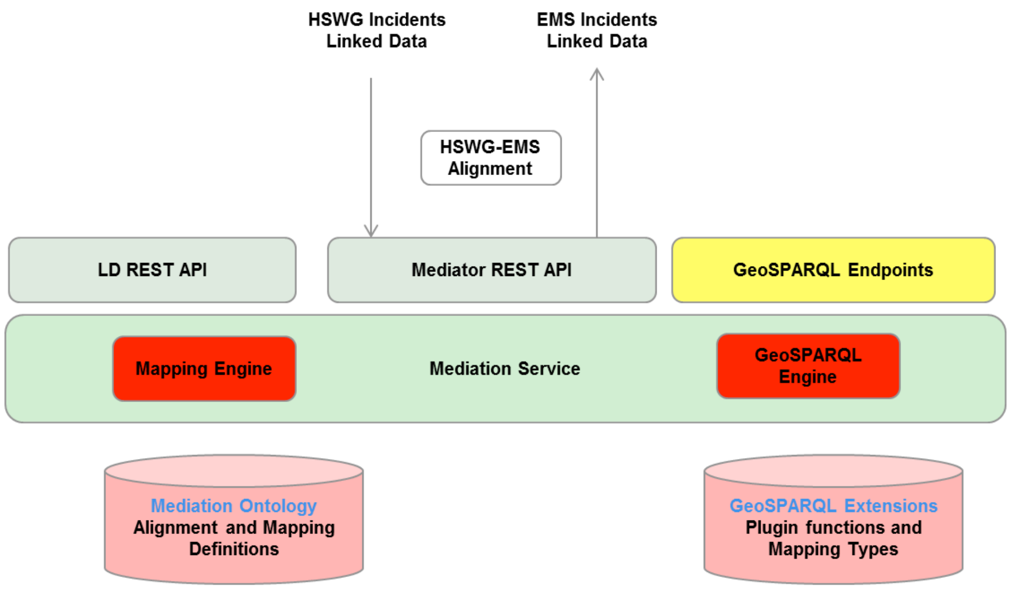


Figure 12 Semantic Mediation Service Architecture

12.1.2 REST API Overview

Table 25 summarizes the REST API implemented for the SMS for this testbed.

Table 25 Semantic Mediation Service REST API Summary

Endpoint	Method	Description	Format

/functions	GET	Get all the functions currently registered with the service.	RDF,TTL, N3,JSON-LD
/mappings/types	GET	Get all the mappings types available in the service	RDF,TTL, N3,JSON-LD
/alignments/model	GET	Get the model containing all the alignments, mappings and supporting taxonomies	RDF,TTL, N3,JSON-LD
/alignments/sparql	GET	SPARQL endpoint to query the alignment model	RDF,XML ,JSON-LD, CSV, TSV, SPARQL-RESULTS XML and JSON
/alignments/instances	GET	List all the instances of Alignments available	RDF,TTL, N3,JSON-LD
/alignments/instances/{name}	GET	Get the alignment with the given name	RDF,TTL, N3,JSON-LD
/alignments/instances/{name}/mediator	GET POST	Mediate a model using the alignment with the given name	RDF,TTL, N3,JSON-LD

12.1.3 Endpoint: /functions

Description: Get all the functions currently registered in the system (based on SPARQL extensions vocabulary). The functions can be filtered by name using the label parameter. The model can be returned in TTL, N3, RDF/XML or JSON-LD formats (using content negotiation or file extension) using the SPARQL Extension ontology.

12.1.3.1 Request

HTTP Method: **GET**

Table 26 summarizes the query parameters accepted by the endpoint.

Table 26 Query Parameters for /functions endpoint

Name	Definition	Type	Multiplicity
label	name of the function	string	0..1

12.1.3.2 Response

The response returns functions defined in SPARQL Extension ontology in TTL, RDF/XML, N3 and JSON-LD.

12.1.3.3 Examples

To get all the functions available in the service, the following request can be done:

http://ows.usersmarts.com/mediator/functions (default Turtle format).

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sd:    <http://www.w3.org/ns/sparql-service-description#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sparql-ext:
<http://www.opengis.net/testbed11/ont/geosparql/extensions#> .

<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#skos
Match>
        a                        owl:Class , sparql-ext:Function ;
        rdfs:comment             "Get the closest match for a given
concept in a target concept scheme. " ;
        rdfs:label               "skosMatch" ;
        rdfs:subClassOf          sparql-ext:FunctionCall ;
        sparql-ext:body          [ a                           sparql-
ext:Select , sparql-ext:Query ;
                                   sparql-ext:queryLanguage
sd:SPARQL11Query ;
                                   sparql-ext:textForm       "PREFIX
geosparql:<http://www.opengis.net/testbed/11/def/function/geosparql/>\r
\n\r\nSELECT ?x\r\nWHERE {\r\n  BIND(geosparql:skosMatch(?arg1,?arg2)
as ?x)\r\n}"
                                 ] ;
        sparql-ext:hasParameter  [ a                       sparql-
ext:Parameter ;
                                   rdfs:comment            "The target
concept scheme" ;
```

```
                                  rdfs:label            "arg2" ;
                                  sparql-ext:predicate  sparql-
ext:arg2 ;
                                  sparql-ext:valueType  rdfs:Resource
                                ] ;
        sparql-ext:hasParameter  [ a                    sparql-
ext:Parameter ;
                                  rdfs:comment          "the concept
to match" ;
                                  rdfs:label            "arg1" ;
                                  sparql-ext:predicate  sparql-
ext:arg1 ;
                                  sparql-ext:valueType  rdfs:Resource
                                ] ;
        sparql-ext:returnType    rdfs:Resource .

<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#URIB
uild2>
        a                        owl:Class , sparql-ext:Function ;
        rdfs:comment             "Builds a new URI using the value of
two given properties (?arg1, ?arg1) from a given subject (?source) and
a given template (?template). The template may reference the values of
the properties using {?1} and {?2}." ;
        rdfs:label               "URIBuild2" ;
        rdfs:subClassOf          sparql-ext:FunctionCall ;
        sparql-ext:body          [ a                        sparql-
ext:Select , sparql-ext:Query ;
                                  sparql-ext:queryLanguage
sd:SPARQL11Query ;
                                  sparql-ext:textForm       "PREFIX
fn:<http://www.usersmarts.com/ont/2005/06/ks/functor#>\r\nSELECT
?uri\r\nWHERE {\r\n  ?source ?arg1 ?value1.\r\n  ?source ?arg2
?value2.\r\n  BIND (URI(fn:substitute(?template,?value1,?value2)) AS
?uri) .\r\n}"
                                ] ;
        sparql-ext:hasParameter  [ a                    sparql-
ext:Parameter ;
                                  rdfs:comment          "Second
predicate of the given subject to select" ;
                                  rdfs:label            "arg1" ;
                                  sparql-ext:predicate  sparql-
ext:arg1 ;
                                  sparql-ext:valueType  rdf:Property
                                ] ;
        sparql-ext:hasParameter  [ a                    sparql-
ext:Parameter ;
                                  rdfs:label            "arg2" ;
                                  sparql-ext:predicate  sparql-
ext:arg2 ;
                                  sparql-ext:valueType  rdf:Property
                                ] ;
        sparql-ext:hasParameter  [ a                    sparql-
ext:Parameter ;
                                  rdfs:comment          "Subject
Resource" ;
```



```
                                   rdfs:label            "source" ;
                                   sparql-ext:predicate
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#sour
ce> ;
                                   sparql-ext:valueType  xsd:string
                                 ] ;
        sparql-ext:hasParameter  [ a                     sparql-
ext:Parameter ;
                                   rdfs:comment          "Template to
use construct the new uri" ;
                                   rdfs:label            "template" ;
                                   sparql-ext:predicate
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#temp
late> ;
                                   sparql-ext:valueType  xsd:string
                                 ] ;
        sparql-ext:returnType    rdfs:Resource .

<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#URIB
uild1>
        a                        sparql-ext:Function , owl:Class ;
        rdfs:comment             "Builds a new URI using the value of a
given property (?arg1) from a given subject (?source) and a given
template (?template). The template may reference the value of the
property using {?1}." ;
        rdfs:label               "URIBuild1" ;
        rdfs:subClassOf          sparql-ext:FunctionCall ;
        sparql-ext:body          [ a                         sparql-
ext:Select , sparql-ext:Query ;
                                   sparql-ext:queryLanguage
sd:SPARQL11Query ;
                                   sparql-ext:textForm       "PREFIX
fn:<http://www.usersmarts.com/ont/2005/06/ks/functor#>\r\nSELECT
?uri\r\nWHERE {\r\n  ?source ?arg1 ?value.\r\n  BIND
(URI(fn:substitute(?template,?value)) AS ?uri) .\r\n}"
                                 ] ;
        sparql-ext:hasParameter  [ a                     sparql-
ext:Parameter ;
                                   rdfs:comment          "predicate of
the given subject to select" ;
                                   rdfs:label            "arg1" ;
                                   sparql-ext:predicate
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#arg1
> ;
                                   sparql-ext:valueType  rdf:Property
                                 ] ;
        sparql-ext:hasParameter  [ a                     sparql-
ext:Parameter ;
                                   rdfs:comment          "Template to
use construct the new uri" ;
                                   rdfs:label            "template" ;
                                   sparql-ext:predicate
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#temp
late> ;
                                   sparql-ext:valueType  xsd:string
                                 ] ;
```

```
        sparql-ext:hasParameter  [ a                      sparql-
ext:Parameter ;
                                   rdfs:comment           "Subject
Resource" ;
                                   rdfs:label             "source" ;
                                   sparql-ext:predicate
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#sour
ce> ;
                                   sparql-ext:valueType  xsd:string
                                 ] ;
        sparql-ext:returnType    rdfs:Resource .

<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#obje
ct>
        a                        owl:Class , sparql-ext:Function ;
        rdfs:comment             "Gets the object of a given subject
(?arg1) / predicate (?arg2) combination. Note that if multiple values
are present then the result might be non deterministic." ;
        rdfs:label               "object" ;
        rdfs:subClassOf          sparql-ext:FunctionCall ;
        sparql-ext:body          [ a                          sparql-
ext:Select , sparql-ext:Query ;
                                   sparql-ext:queryLanguage
sd:SPARQL11Query ;
                                   sparql-ext:textForm        "SELECT
?object\r\nWHERE {\r\n  ?arg1 ?arg2 ?object.\r\n}"
                                 ] ;
        sparql-ext:hasParameter  [ a                      sparql-
ext:Parameter ;
                                   rdfs:comment           "The predicate
to get the object of." ;
                                   rdfs:label             "arg2" ;
                                   sparql-ext:predicate  sparql-
ext:arg2 ;
                                   sparql-ext:valueType  rdf:Property
                                 ] ;
        sparql-ext:hasParameter  [ a                      sparql-
ext:Parameter ;
                                   rdfs:comment           "The subject
to get the object from." ;
                                   rdfs:label             "arg1" ;
                                   sparql-ext:predicate  sparql-
ext:arg1 ;
                                   sparql-ext:valueType  rdfs:Resource
                                 ] ;
        sparql-ext:returnType    rdfs:Resource .

<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#Chan
geNamespace>
        a                        owl:Class , sparql-ext:Function ;
        rdfs:comment             "Function changing the namespace of a
uri" ;
        rdfs:label               "ChangeNamespace" ;
        rdfs:subClassOf          sparql-ext:FunctionCall ;
```



```
        sparql-ext:body          [ a                        sparql-
ext:Select , sparql-ext:Query ;
                                   sparql-ext:queryLanguage
sd:SPARQL11Query ;
                                   sparql-ext:textForm       "PREFIX
fn:<http://www.usersmarts.com/ont/2005/06/ks/functor#>\r\nSELECT
?uri\r\nWHERE {\r\n  BIND ( URI(
CONCAT(?targetNamespace,fn:localName(?arg1)) ) AS ?uri) .\r\n}"
                                 ] ;
        sparql-ext:hasParameter  [ a                    sparql-
ext:Parameter ;
                                   rdfs:comment         "Target
namespace to substitute" ;
                                   rdfs:label
"targetNamespace" ;
                                   sparql-ext:predicate
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#targ
etNamespace> ;
                                   sparql-ext:valueType  rdfs:Resource
                                 ] ;
        sparql-ext:hasParameter  [ a                    sparql-
ext:Parameter ;
                                   rdfs:comment         "URI Resource
to change namespace" ;
                                   rdfs:label           "arg1" ;
                                   sparql-ext:predicate  sparql-
ext:arg1 ;
                                   sparql-ext:valueType  xsd:string
                                 ] ;
        sparql-ext:returnType    rdfs:Resource .
```

The functions can be searched by label using the label parameter. For example to get the function named skosMatch, the request looks like this:
http://ows.usersmarts.com/mediator/functions?label=skosMatch

The response in TTL is the following:

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sd:    <http://www.w3.org/ns/sparql-service-description#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sparql-ext:
<http://www.opengis.net/testbed11/ont/geosparql/extensions#> .

<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#skos
Match>
        a                        owl:Class , sparql-ext:Function ;
        rdfs:comment             "Get the closest match for a given
concept in a target concept scheme. " ;
        rdfs:label               "skosMatch" ;
        rdfs:subClassOf          sparql-ext:FunctionCall ;
```



```
        sparql-ext:body          [ a                          sparql-
ext:Select , sparql-ext:Query ;
                                   sparql-ext:queryLanguage
sd:SPARQL11Query ;
                                   sparql-ext:textForm        "PREFIX
geosparql:<http://www.opengis.net/testbed/11/def/function/geosparql/>\r
\n\r\nSELECT ?x\r\nWHERE {\r\n  BIND(geosparql:skosMatch(?arg1,?arg2)
as ?x)\r\n}"
                                 ] ;
        sparql-ext:hasParameter  [ a                      sparql-
ext:Parameter ;
                                   rdfs:comment           "The target
concept scheme" ;
                                   rdfs:label             "arg2" ;
                                   sparql-ext:predicate   sparql-
ext:arg2 ;
                                   sparql-ext:valueType   rdfs:Resource
                                 ] ;
        sparql-ext:hasParameter  [ a                      sparql-
ext:Parameter ;
                                   rdfs:comment           "the concept
to match" ;
                                   rdfs:label             "arg1" ;
                                   sparql-ext:predicate   sparql-
ext:arg1 ;
                                   sparql-ext:valueType   rdfs:Resource
                                 ] ;
        sparql-ext:returnType    rdfs:Resource .
```

12.1.4 Endpoint: /mappings/types

Description: Get all the mapping types currently registered in the system (based on SPARQL extensions vocabulary). The model can be returned in TTL, N3, RDF/XML or JSON-lD formats (using content negotiation or file extension).

12.1.4.1 Request

HTTP Method: **GET**

Table 27 summarizes the query parameters accepted by the endpoint.

Table 27 Query parameters for /mappings/types endpoint

Name	**Definition**	**Type**	**Multiplicity**
label	name of the function	string	0..1

12.1.4.2 Response

12.1.4.3 Examples

http://ows.usersmarts.com/mediator/mappings/types (default Turtle format).

http://ows.usersmarts.com/mediator/mappings/types.jsonld (model in JSON-LD using jsonld file extension).

The mapping types can be searched by label using the label parameter: http://ows.usersmarts.com/mediator/mappings/types?label=ClassMapping

12.1.5 Endpoint: /alignments/model

Description: Get the model contains all the alignments defined in the registry. The model can be returned in TTL, N3, RDF/XML or JSON-lD formats (using content negotiation or file extension). Note the current versions merge the alignments with the taxonomies used to perform taxonomy mediation (using **skosMatch** functions).

12.1.5.1 Request

HTTP Method: GET

This method does not take any query parameters.

12.1.5.2 Response

The response returns the RDF model containing all the alignments definitions including their supporting taxonomies and mappings. The model can be returned in TTL, N3, RDF/XML or JSON-LD formats (using content negotiation or file extension).

12.1.5.3 Example

http://ows.usersmarts.com/mediator/alignments/model

12.1.6 Endpoint: /alignments/sparql

Description: SPARQL endpoint performing search on the model containing the alignments

12.1.6.1 Request

The query parameters for the endpoint are summarized in Table 28 . They are aligned with standard SPARQL protocol. The SPARQL query is executed against the alignment model.

Table 28 Query parameters for /alignments/sparql endpoint

Name	Definition	Type	Multiplicity

query	SPARQL query to execute against the model	string	1

12.1.6.2 Response

The response format depends of the types of query. SPARQL-Results in XML and JSON format , CSV, TSV, RDF,TTL, N3,JSON-LD are supported.

12.1.6.3 Example

http://ows.usersmarts.com/mediator/alignments/sparql

12.1.7 Endpoint: /alignments/instances

Description: List the instances of the alignment available (at present only one for HSWG to EMS)

12.1.7.1 Request

HTTP Method: GET

Table 29 summarizes the query parameters supported by the endpoint.

Table 29 Query Parameters for /alignments/instances endpoint

Name	Definition	Type	Multiplicity
srcOntology	The url of the source ontology	url (encoded)	0..1
targetOntology	The url of the target ontology	url (encoded)	0..1
name	name of the alignment to search	String	0..1

12.1.7.2 Response

The response returns instances of **mediation:Alignment** in RDF/XML, TTL, N3 and JSON-LD format.

12.1.7.3 Examples

The following query returns all the alignment instances supported by the service.

http://ows.usersmarts.com/mediator/alignments/instances

The following query finds the alignments from the HSWG Incident ontology to the target EMS Incident ontology. The URLs of the ontologies are URL encoded in the query.

http://ows.usersmarts.com/mediator/alignments/instances?srcOntology=http%3A%2F%2Fwww.opengis.net%2Ftestbed11%2Font%2Fincident%2Fhswg%23&targetOntology=http%3A%2F%2Fwww.opengis.net%2Ftestbed11%2Font%2Fincident%2Fems%23

The following query finds the alignment with the name equals to HSWG2EMS.

http://ows.usersmarts.com/mediator/alignments/instances?name=HSWG2EMS

The response of all three requests returns the same representation in TTL format (as only one alignment was implemented for the testbed).

```
@base          <http://www.opengis.net/taxonomy/ems#> .
@prefix ogc-map:
<http://www.opengis.net/testbed11/ont/geosparql/mapping/core#> .
@prefix sd:    <http://www.w3.org/ns/sparql-service-description#> .
@prefix natural-events: <http://www.fgdc.gov/HSWG/taxonomy/natural-
events#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix ems:   <http://www.opengis.net/taxonomy/ems#> .
@prefix lda:   <http://www.knowledgesmarts.com/ontologies/lda#> .
@prefix incidents: <http://www.fgdc.gov/HSWG/taxonomy/incidents#> .
@prefix geosparql-fn:
<http://www.opengis.net/testbed/11/def/function/geosparql/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix mediation:
<http://www.opengis.net/testbed11/ont/geosparql/mediation#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix spin:  <http://spinrdf.org/spin#> .
@prefix sparql-ext:
<http://www.opengis.net/testbed11/ont/geosparql/extensions#> .
@prefix fn:
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix incident: <http://www.opengis.net/ont/emergency/incident#> .

<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#HSWG-EMS-
Alignment>
        a                        mediation:Alignment ;
        rdfs:comment             "Alignment between HSWG Incident and
EMS Incident" ;
        rdfs:label               "Alignment between HSWG Incident and
EMS Incident" ;
        mediation:alignmentName  "HSWG2EMS" ;
        mediation:hasMapping
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#city-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#asWKT-
```



```
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#date-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#hasAddress
-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#AddressMap
ping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#latitude-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#longitude-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#incidentId
-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#fullAddres
s-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#incidentTy
pe-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#label-
title-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#descriptio
n-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#PointMappi
ng> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#time-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#HSWG-EMS-
IncidentMapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#state-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#hasPositio
n-mapping> ;
        mediation:sourceOntology
<http://www.opengis.net/testbed11/ont/incident/hswg#> ;
        mediation:targetOntology
<http://www.opengis.net/testbed11/ont/incident/ems#> .
```

12.1.8 Endpoint: /alignments/instances/{id}

Description: Get the description of the alignment identified with the name **id**. The name is used in the URL template for the REST API to access the instance of the alignment

12.1.8.1 Request

HTTP Method: Get

This request does not accept any parameters.

12.1.8.2 Response

The response returns the instance of the alignment with the name equals to the **id** parameter in the URL pattern. The response can be returned in RDF/XML, TTL, N3 and JSON-LD.

If the instance is not found, a HTTP code **404** is returned with the following JSON response


```
{
    "reason": "Not Found",
    "status": 404,
    "description": "Resource does not exist",
    "applicationName": "Semantic Mediation Service"
}
```


12.1.8.3 Example

http://ows.usersmarts.com/mediator/alignments/instances/HSWG2EMS

The response in TTL is the following:

```
@base          <http://www.opengis.net/taxonomy/ems#> .
@prefix ogc-map:
<http://www.opengis.net/testbed11/ont/geosparql/mapping/core#> .
@prefix sd:    <http://www.w3.org/ns/sparql-service-description#> .
@prefix natural-events: <http://www.fgdc.gov/HSWG/taxonomy/natural-
events#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix ems:   <http://www.opengis.net/taxonomy/ems#> .
@prefix lda:   <http://www.knowledgesmarts.com/ontologies/lda#> .
@prefix incidents: <http://www.fgdc.gov/HSWG/taxonomy/incidents#> .
@prefix geosparql-fn:
<http://www.opengis.net/testbed/11/def/function/geosparql/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix mediation:
<http://www.opengis.net/testbed11/ont/geosparql/mediation#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix spin:  <http://spinrdf.org/spin#> .
@prefix sparql-ext:
<http://www.opengis.net/testbed11/ont/geosparql/extensions#> .
@prefix fn:
<http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix incident: <http://www.opengis.net/ont/emergency/incident#> .
```

```
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#HSWG-EMS-
Alignment>
        a                         mediation:Alignment ;
        rdfs:comment              "Alignment between HSWG Incident and
EMS Incident" ;
        rdfs:label                "Alignment between HSWG Incident and
EMS Incident" ;
        mediation:alignmentName   "HSWG2EMS" ;
        mediation:hasMapping
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#city-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#asWKT-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#date-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#hasAddress
-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#AddressMap
ping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#latitude-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#longitude-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#incidentId
-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#fullAddres
s-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#incidentTy
pe-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#label-
title-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#descriptio
n-mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#PointMappi
ng> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#time-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#HSWG-EMS-
IncidentMapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#state-
mapping> ,
<http://ows.usersmarts.com/mediator/mappings/HSWG2EMSMapping#hasPositio
n-mapping> ;
        mediation:sourceOntology
<http://www.opengis.net/testbed11/ont/incident/hswg#> ;
        mediation:targetOntology
<http://www.opengis.net/testbed11/ont/incident/ems#> .
```

12.1.9 Endpoint: /alignments/instances/{id}/mediator

Description: This endpoint performs the mediation of a Linked Data Model using the alignment identified the given identifier (id). They are two supported methods : using HTTP Get when models can accessed using a URL and using POST when a model serialized in RDF,TTL, JSON-LD or N3 are submitted to perform the mediation. In both cases, the mediation service returns the transformed model in the target ontology defined in the alignment.

12.1.9.1 HTTP Get Request

Description: The HTTP GET takes at present a reference to the model contains data in the source ontology (in our case HSWG Incident). The parameter referring to the model is **srcModel.** Table 30 summarizes the query parameters of this endpoint.

Table 30 Query parameters for /alignments/{id}/mediator endpoint

Name	Definition	Type	Multiplicity
srcModel	URL of the model expressed in the source ontology of the alignment.	URL	1

12.1.9.2 Response

The response returns the transformed model using the mappings of the alignment with the given identifier id. The output can be returned in RDF/XML, TTL, JSON-LD or N3.

12.1.9.3 Example

If you want to transform for example the following Incident located at the following REST endpoint: http://ows.usersmarts.com/ldapp/ows11/demo/ems/sfpd/incidents/11616059406244, here the call to the REST endpoint (url encode the URL).

http://ows.usersmarts.com/mediator/alignments/instances/HSWG2EMS/mediator?srcModel=http%3A%2F%2Fows.usersmarts.com%2Fldapp%2Fows11%2Fdemo%2Fems%2Fsfpd%2Fincidents%2F11616059406244

The input data is:

@prefix ks: <http://www.usersmarts.com/ont/2005/06/ks#> .
@prefix spatial: <http://www.opengis.net/ont/spatial#> .
@prefix hswg: <http://www.opengis.net/testbed11/ont/incident/hswg#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix time: <http://www.knowledgesmarts.com/ontology/time#> .
@prefix evt: <http://www.knowledgesmarts.com/ontologies/event#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

```
@prefix place: <http://www.knowledgesmarts.com/ontologies/place#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix evt-type: <http://www.smartrealm.com/ct/types/events#> .
@prefix ptype: <http://www.knowledgesmarts.com/ontologies/place/types#> .
@prefix incident: <http://www.opengis.net/ont/domain/emergency/police/incident#> .

<http://ows.usersmarts.com/ldapp/ows11/demo/ems/sfpd/incidents/11616059406244>
      a                  hswg:HSWGIncident ;
      hswg:hasAddress      [ a                  hswg:Address ;
                        hswg:city           "San Francisco" ;
                        hswg:fullAddress     "SHERIDAN ST / 9TH ST" ;
                        hswg:policeDistrict  "SOUTHERN" ;
                        hswg:state           "CA"
                      ] ;
      hswg:incidentDate    "2011-12-11"^^xsd:date ;
      hswg:incidentNumber  "116160594" ;
      hswg:incidentTime    "03:00:00"^^xsd:time ;
      hswg:incidentType    <http://www.fgdc.gov/HSWG/taxonomy/incidents#Looting> ;
      hswg:location        [ a                wgs84:Point , geosparql:Point ;
                        geosparql:asWKT  "POINT (-122.4106935
37.77302471)"^^geosparql:wktLiteral ;
                        wgs84:lat        37.77302471 ;
                        wgs84:long       -122.4106935
                      ] ;
      hswg:resolution      "NONE" ;
      hswg:summary         "GRAND THEFT FROM LOCKED AUTO" .
```

The source model is processed by the mediator by applying recursively the mappings associated with the alignment to produce the transformed results.

```
@base          <http://www.opengis.net/taxonomy/ems#> .
@prefix ogc-map: <http://www.opengis.net/testbed11/ont/geosparql/mapping/core#> .
@prefix sd:    <http://www.w3.org/ns/sparql-service-description#> .
@prefix natural-events: <http://www.fgdc.gov/HSWG/taxonomy/natural-events#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix ems:   <http://www.opengis.net/taxonomy/ems#> .
@prefix lda:   <http://www.knowledgesmarts.com/ontologies/lda#> .
@prefix incidents: <http://www.fgdc.gov/HSWG/taxonomy/incidents#> .
@prefix geosparql-fn: <http://www.opengis.net/testbed/11/def/function/geosparql/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix mediation: <http://www.opengis.net/testbed11/ont/geosparql/mediation#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
```



```
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix spin:  <http://spinrdf.org/spin#> .
@prefix sparql-ext: <http://www.opengis.net/testbed11/ont/geosparql/extensions#> .
@prefix fn:    <http://www.opengis.net/testbed11/ont/geosparql/ext/functions/core#> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .
@prefix incident: <http://www.opengis.net/ont/emergency/incident#> .

<http://ows.usersmarts.com/testbed11/data/ems#11616059406244>
      a       <http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident> ;
      <http://www.opengis.net/testbed11/ont/incident/ems#address>
              [ a                 vcard:Address ;
                vcard:locality      "San Francisco" ;
                vcard:region        "CA" ;
                vcard:street-address  "SHERIDAN ST / 9TH ST"
              ] ;
      <http://www.opengis.net/testbed11/ont/incident/ems#description>
              "GRAND THEFT FROM LOCKED AUTO" ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentDate>
              "2011-12-11"^^xsd:date ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentId>
              "116160594" ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentTime>
              "03:00:00"^^xsd:time ;
      <http://www.opengis.net/testbed11/ont/incident/ems#incidentType>
              ems:ems.incident.crime.looting ;
      <http://www.opengis.net/testbed11/ont/incident/ems#position>
              [ a             geosparql:Point ;
                geosparql:asWKT  "POINT (-122.4106935
37.77302471)"^^geosparql:wktLiteral ;
                wgs84:lat       37.77302471 ;
                wgs84:long      -122.4106935
              ] .
```

Note the transformation on the identifier of the incident by changing the namespace, the change of structure of the Address (using VCard vocabulary) and the taxonomy mediation from HSWG to EMS (http://www.fgdc.gov/HSWG/taxonomy/incidents#Looting is **skos:exactMatch** to ems:ems.incident.crime.looting). The Mapping engine can perform transformations on multiple incidents even with partial information (such Address or Point).

12.1.9.4 HTTP Post Request

To use this request, simply post a RDF document in TTL, RDF, N3 in the body of the POST to the endpoint of the alignment mediator and the service will return the transformed model to the target ontology.

12.2 Image Matters Semantic Portrayal Service

Image Matters deployed an initial version of semantic portrayal service (also known as symbology service) online at the following endpoint: http://ows.usersmarts.com/portrayal/api

The server was loaded with the EMS, HSWG symbols, taxonomies and portrayal rules produced during the testbed. The service has also an initial REST API to fetch symbols and symbol sets in TTL, RDF/XML, JSON-LD. N3 and NT formats.

12.2.1 Architecture overview

The server consists of a standard RDF database (Systap BlazeGraph) where all the portrayal information was stored as Linked Data. A REST API was built on top of the repository.

12.2.2 REST API Overview

Table 31 summarizes the Portrayal Service REST API. The main endpoint is the SPARQL endpoint allowing access to any portrayal information (style, portrayal rules, symbol sets, symbols, graphics and supporting taxonomies). Two other endpoints provide a Linked Data API to access symbol sets and symbols. Future extensions will provide access to styles, portrayal rules and graphics in a RESTful way.

Table 31 Portrayal Service REST API Summary

Endpoint	Method	Description	Format
/symbolsets	GET	Get collection of symbolSets	RDF,TTL, N3,JSON-LD
/symbols	GET	Get collection of symbols	RDF,TTL, N3,JSON-LD
/sparql	GET	SPARQL endpoint to query the portrayal	RDF,XML ,JSON-LD, CSV,

		information	TSV, SPARQL-RESULTS XML and JSON

12.2.3 Endpoint: /symbolsets

Description: Returns the collection symbol sets available in the service

12.2.3.1 Request

HTTP Method: Get

No parameters are supported by the request.

12.2.3.2 Response

The response returns Linked Data representation of the symbol sets according the Symbology ontology in TTL, RDF/XML, N3 and JSON-LD.

12.2.3.3 Example

The following endpoint http://ows.usersmarts.com/portrayal/api/symbolsets returns all the symbol sets available in the service. In this case two symbol sets are returned HSWG and EMS with references to all the symbols they contain.

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix symbol: <http://www.opengis.net/ont/portrayal/symbol#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .

<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet>
        a                     symbol:SymbolSet ;
        dct:description       "Standard Canadian Emergency Mapping
Symbology (EMS) SymbolSet version 1.0" ;
        dct:title             "Canadian Emergency Mapping Symbology
(EMS) SymbolSet (version 1.0)" ;
        symbol:hasSymbol
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.airQual
ity-symbol> ,
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth-symbol> ,
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth.animalDieOff-symbol> ,(truncated)
        symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> .

<http://www.opengis.net/testbed/11/cci/hswg/symbols#HSWGSymbolSet>
```

```
        a                      symbol:SymbolSet ;
        dct:description        "Home Security Working Group (HSWG)
SymbolSet version 1.0" ;
        dct:title              "HSWG SymbolSet (version 1.0)" ;
        symbol:hasSymbol
<http://www.opengis.net/testbed/11/cci/hswg/symbols#AirAccidentSymbol>,
<http://www.opengis.net/testbed/11/cci/hswg/symbols#AirHijackingSymbol>
,<http://www.opengis.net/testbed/11/cci/hswg/symbols#AirIncidentSymbol>
(truncated)
symbol:specification
<http://www.fgdc.gov/HSWG/ref_pages/Incidents_ref.htm> .
```

12.2.4 Endpoint: /symbols

Description: Returns the collection symbols available in the service based on the parameters of the query. The symbols can be filtered by symbol sets or by given a list of symbols URI explicitly. The response is returned as Linked Data according the Symbology ontology.

12.2.4.1 Request

HTTP Method: GET

Table 32 summarizes the query parameters for this endpoint.

Table 32 Query Parameters of the /symbols endpoint

Name	Definition	Type	Multiplicity
symbolSetURI	The URI of the symbol sets from which the symbols are members	URI (encoded)	0..1
uri	URI of the symbol to fetch	URI (encoded)	0..n

12.2.4.2 Response

The response returns Linked Data representation of the symbols according the Symbology ontology in TTL, RDF/XML, N3 and JSON-LD.

12.2.4.3 Examples

The following endpoint http://ows.usersmarts.com/portrayal/api/symbols gives the list of all symbols available within the service (HSWG and EMS in this testbed).

To get the symbols for EMS symbolSet, the request looks like:

http://ows.usersmarts.com/portrayal/api/symbols?symbolSetURI=http%3A%2F%2Fwww.opengis.net%2Ftestbed%2F11%2Fcci%2Fems%2Fsymbols%23EMSSymbolSet

A sample of the response in TTL follows:

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix symbol: <http://www.opengis.net/ont/portrayal/symbol#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .

<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.airQual
ity-symbol>
        a                          symbol:Symbol ;
        rdfs:label                 "airQuality" ;
        dct:identifier             "ems.incident.airQuality" ;
        symbol:definition
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.airQual
ity-symbolDefinition> ;
        symbol:denotes
<http://www.opengis.net/taxonomy/ems#ems.incident.airQuality> ;
        symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> ;
        symbol:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet> ;
        skos:notation
"ems.incident.airQuality"^^<http://www.opengis.net/testbed/11/cci/ems/s
ymbols#emsNotation> .

<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth-symbol>
        a                          symbol:Symbol ;
        rdfs:label                 "animalHealth" ;
        dct:identifier             "ems.incident.animalHealth" ;
        symbol:definition
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth-symbolDefinition> ;
        symbol:denotes
<http://www.opengis.net/taxonomy/ems#ems.incident.animalHealth> ;
        symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> ;
        symbol:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet> ;
        skos:notation
"ems.incident.animalHealth"^^<http://www.opengis.net/testbed/11/cci/ems
/symbols#emsNotation> .

<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth.animalDieOff-symbol>
        a                          symbol:Symbol ;
        rdfs:label                 "animalDieOff" ;
        dct:identifier             "ems.incident.animalHealth.animalDieOff"
;
```

```
        symbol:definition
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth.animalDieOff-symbolDefinition> ;
        symbol:denotes
<http://www.opengis.net/taxonomy/ems#ems.incident.animalHealth.animalDi
eOff> ;
        symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> ;
        symbol:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet> ;
        skos:notation
"ems.incident.animalHealth.animalDieOff"^^<http://www.opengis.net/testb
ed/11/cci/ems/symbols#emsNotation> .
…..
```

Here an example to fetch two symbols descriptions

http://ows.usersmarts.com/portrayal/api/symbols?uri=http%3A%2F%2Fwww.opengis.net%2Ftestbed%2F11%2Fcci%2Fems%2Fsymbols%23ems.incident.airQuality-symbol&uri=http%3A%2F%2Fwww.opengis.net%2Ftestbed%2F11%2Fcci%2Fems%2Fsymbols%23ems.incident.animalHealth-symbol

The response in TTL is the following:

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix symbol: <http://www.opengis.net/ont/portrayal/symbol#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .

<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.airQual
ity-symbol>
        a                      symbol:Symbol ;
        rdfs:label             "airQuality" ;
        dct:identifier         "ems.incident.airQuality" ;
        symbol:definition
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.airQual
ity-symbolDefinition> ;
        symbol:denotes
<http://www.opengis.net/taxonomy/ems#ems.incident.airQuality> ;
        symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> ;
        symbol:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet> ;
        skos:notation
"ems.incident.airQuality"^^<http://www.opengis.net/testbed/11/cci/ems/s
ymbols#emsNotation> .

<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth-symbol>
        a                      symbol:Symbol ;
        rdfs:label             "animalHealth" ;
```



```
        dct:identifier        "ems.incident.animalHealth" ;
        symbol:definition
<http://www.opengis.net/testbed/11/cci/ems/symbols#ems.incident.animalH
ealth-symbolDefinition> ;
        symbol:denotes
<http://www.opengis.net/taxonomy/ems#ems.incident.animalHealth> ;
        symbol:specification  <https://cms.masas-
x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> ;
        symbol:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet> ;
        skos:notation
"ems.incident.animalHealth"^^<http://www.opengis.net/testbed/11/cci/ems
/symbols#emsNotation> .
```

12.2.5 Endpoint: /sparql

Description: The service provides a standard SPARQL endpoint (based on W3C SPARQL protocol) to query the model containing all the portrayal information managed by the server.

12.2.5.1 Request

HTTP Method: GET

The query parameters for the endpoint are summarized in Table 33. They are aligned with standard SPARQL protocol. The SPARQL query is executed against the alignment model.

Table 33 Query parameters for /sparql endpoint

Name	Definition	Type	Multiplicity
query	SPARQL query to execute against the portrayal service model	string	1

12.2.5.2 Response

The response format depends of the types of query. SPARQL-Results in XML and JSON format, CSV, TSV, RDF,TTL, N3,JSON-LD are supported.

12.2.5.3 Examples

A SPARQL client was provided at http://ows.usersmarts.com/portrayal/api/sparql to test different queries. The endpoint was used programmatically by the Geomatys WPS to generate SLD documents from a given Style. Queries were sent under the following form:

http://ows.usersmarts.com/portrayal/api/sparql?query=your encoded sparql query

Here some sample queries that could be sent to the server:

The following query describes the list of SymbolSets available in the knowledge base

```
PREFIX symbol:<http://www.opengis.net/ont/portrayal/symbol#>

DESCRIBE ?symbolSet WHERE {
  ?symbolSet a  symbol:SymbolSet.
}
```

The following query lists the symbols from EMS with label and notation

```
PREFIX symbol:<http://www.opengis.net/ont/portrayal/symbol#>
PREFIX dct:   <http://purl.org/dc/terms/>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?symbol ?label ?notation WHERE {
  ?symbol a symbol:Symbol;
          rdfs:label ?label;
          skos:notation ?notation;
          symbol:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet>.
}
```

Figure 13 SPARQL Client response shows the response in the SPARQL client.

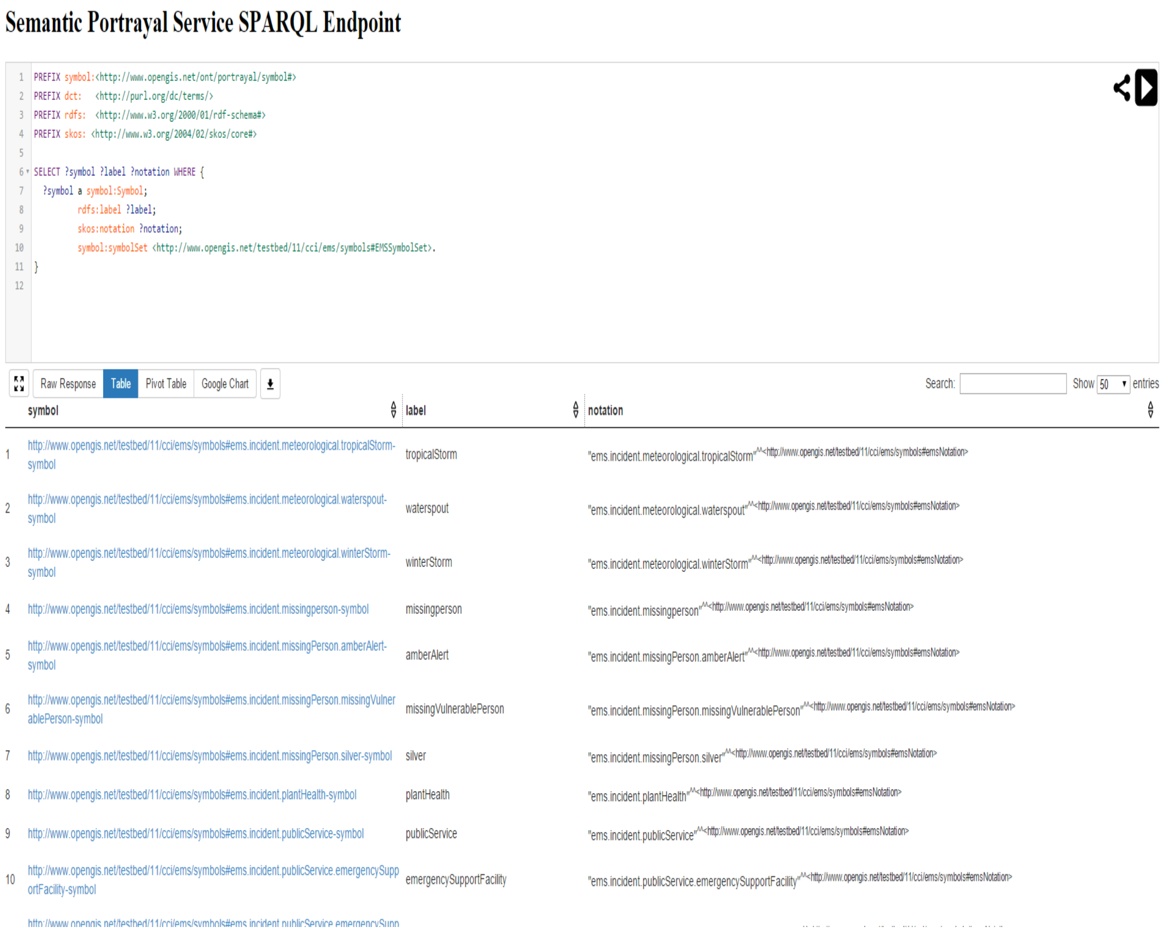


Figure 13 SPARQL Client response

Describe the styles available in the knowledge base

```
PREFIX style:<http://www.opengis.net/ont/portrayal/style#>

DESCRIBE  ?style {
  ?style a style:Style.
}
```

The following response in TTL is returned

```
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix style: <http://www.opengis.net/ont/portrayal/style#> .
@prefix symbol: <http://www.opengis.net/ont/portrayal/symbol#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix skos:  <http://www.w3.org/2004/02/skos/core#> .

<http://www.opengis.net/testbed/11/cci/ems/style#EMSStyle>
        a               style:Style ;
```

```
        dct:audience
<http://ows.usersmarts.com/ldapp/audiences/community/CanadianEmergencyAndDisast
erManagement> ;
        dct:description   "Style defining the set of rules for mapping incident types from
EMS to symbology" ;
        dct:title         "EMS Style" ;
        style:hasRuleSet  <http://www.opengis.net/testbed/11/cci/ems/style#EMSRuleSet> ;
        style:symbolSet
<http://www.opengis.net/testbed/11/cci/ems/symbols#EMSSymbolSet> .

<http://www.opengis.net/testbed/11/cci/hswg/style#HSWGStyle>
        a                 style:Style ;
        dct:audience
<http://ows.usersmarts.com/ldapp/audiences/community/HSGWEmergencyAndDisaster
Management> ;
        dct:description   "Style defining the set of rules for mapping incident types from
HSWG to symbology" ;
        dct:title         "HSWG Style" ;
        style:hasRuleSet
<http://www.opengis.net/testbed/11/cci/hswg/style#HSWGRuleSet> ;
        style:symbolSet
<http://www.opengis.net/testbed/11/cci/hswg/symbols#HSWGSymbolSet> .
```

Select rules from EMS Style that portrays EMSIncident

```
PREFIX style:<http://www.opengis.net/ont/portrayal/style#>
PREFIX incident:<http://www.opengis.net/ont/emergency/incident#>

SELECT  ?rule {
  <http://www.opengis.net/testbed/11/cci/ems/style#EMSRuleSet> style:hasRule ?rule.
  ?rule a style:PortrayalRule.
  ?rule style:featureType
<http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident>
}
```

The following SPARQL results in JSON is returned


```
{
  "head": {
    "vars": [ "rule" ]
  } ,
  "results": {
    "bindings": [
```


{
"rule": { "type": "uri" , "value":
"http://www.opengis.net/testbed/11/cci/ems/style#ems.incident.meteorological.waterspout-portrayal-rule" }
} ,
{
"rule": { "type": "uri" , "value":
"http://www.opengis.net/testbed/11/cci/ems/style#ems.incident.meteorological.winterStorm-portrayal-rule" }
} ,
{
"rule": { "type": "uri" , "value":
"http://www.opengis.net/testbed/11/cci/ems/style#ems.incident.missingperson-portrayal-rule" }
} ,
…. (truncated)
]
}
}

12.3 Envitia Portrayal Service

Within the symbology mediation thread, Envitia focused on publishing Aviation Symbology Ontologies through a SPARQL Server. During flight, the same aeronautical information can be presented with different colours and other styling depending on whether the information is being viewed in daylight or in the darkness of night. In Testbed 11, the Aviation sub-thread had a requirement to encode symbology styling in an ontology that distinguished day-time symbols from those that are used at night-time. The two groups of symbols were to be represented as separate communities (audience). The sub-thread also had a requirement to publish the ontology through a SPARQL Server. This section describes the approach that was implemented for modelling and publishing the ontology.

The testbed considered how the ontology should encode the values of the different parameters of symbol styles (e.g. fill colour, stroke colour, line thickness and so on). Taking Style Layer Descriptors (SLD) as a case study, the testbed encoded the styling information using a model that included a symbol and a portrayal rule.

An example of how the definition of a symbol was serialised in an RDF document is shown in the following figure. The figure also shows an example of how SLD were serialised within an RDF document describing a portrayal rule as defined in the Style ontology.

```
<http://1-dot-env072015.appspot.com/resource/symbol/AviationDark/RunwayElementType>
        a                 symbol:Symbol , owl:NamedIndividual, foaf:Document ;
        rdfs:label        "RunwayElementType_DARK" ;
        dct:description   "RunwayElementType_DARK symbol" ;
        dct:publisher     "ICAO" ;
        dct:audience      <http://1-dot-
env072015.appspot.com/resource/community/AviationDark> ;
        symbol:portrayalRule  <http://1-dot-
env072015.appspot.com/resource/portrayalrule/AviationDark/RunwayElementType>.

<http://1-dot-
env072015.appspot.com/resource/portrayalrule/AviationDark/RunwayElementType>
        a                  symbol:PortrayalRule , owl:NamedIndividual ;
        dct:description     "RunwayElementType_DARK Rule" ;
        symbol:featureType   "RunwayElementType" ;
        symbol:hasRuleCondition  [ a            symbol:PortrayalRuleCondition ;
                                symbol:hasOGCSLD  [ a        symbol:OGCSLDRule ;
                                                symbol:body  "<Rule xmlns=\"http://www.opengis.net/se\"
xmlns:ns0=\"http://www.aixm.aero/schema/5.1\"
xmlns:ogc=\"http://www.opengis.net/ogc\">
<MinScaleDenominator>0.0</MinScaleDenominator>
<MaxScaleDenominator>300000.0</MaxScaleDenominator><PolygonSymbolizer>
<Geometry><ogc:PropertyName
xmlns:ns0=\"http://www.aixm.aero/schema/5.1\">ns0:timeSlice/ns0:RunwayElementTimeSli
ce</ogc:PropertyName></Geometry><Fill><SvgParameter name=\"fill-opacity\">
<ogc:Literal>1.0</ogc:Literal>       </SvgParameter><SvgParameter
name=\"fill\">#333333</SvgParameter>      </Fill><Stroke><SvgParameter name=\"stroke-
opacity\">          <ogc:Literal>1.0</ogc:Literal></SvgParameter><SvgParameter
name=\"stroke\">#605A4E</SvgParameter><SvgParameter name=\"stroke-width\">
<ogc:Literal>1</ogc:Literal></SvgParameter>      </Stroke></PolygonSymbolizer>
<TextSymbolizer><Geometry><ogc:PropertyName
xmlns:ns0=\"http://www.aixm.aero/schema/5.1\">ns0:timeSlice/ns0:RunwayElementTimeSli
ce</ogc:PropertyName></Geometry><Label><ogc:Function name=\"valueOf\">
<ogc:PropertyName
xmlns:ns0=\"http://www.aixm.aero/schema/5.1\">ns0:timeSlice/ns0:RunwayElementTimeSli
ce/ns0:associatedRunway</ogc:PropertyName><ogc:Literal>          <ogc:PropertyName
xmlns:ns0=\"http://www.aixm.aero/schema/5.1\">ns0:timeSlice/ns0:RunwayTimeSlice/ns0:d
esignator</ogc:PropertyName></ogc:Literal></ogc:Function>      </Label>     <Fill>
<SvgParameter name=\"fill-opacity\">          <ogc:Literal>1.0</ogc:Literal></SvgParameter>
<SvgParameter name=\"fill\"><ogc:Literal>#FFFFFF</ogc:Literal></SvgParameter>
</Fill></TextSymbolizer></Rule>"
                                              ]
                               ] .
```

As illustrated in the figure, each symbol was provided with a label, a description, a literal naming the publisher, a reference to the community of interest (represented as an audience) and a reference to the portrayal rule associated with the symbol. A predicate called symbol:portrayalRule (was designed to reference a portrayal rule from its associated symbol (Errata should be style:portrayalRule).

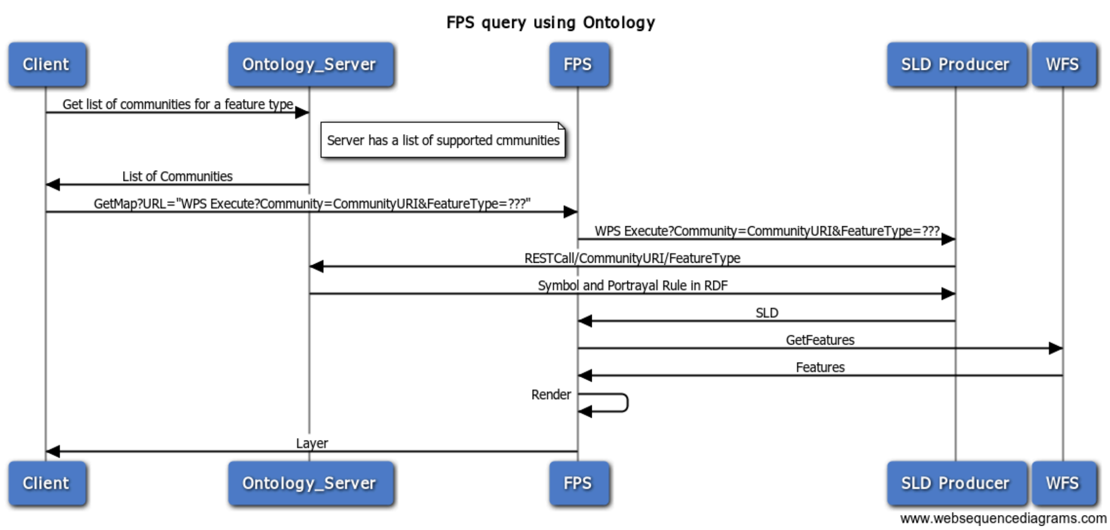
As also illustrated in the figure, the portrayal rule included a reference to the feature type that the styling applied to, as well as the rule condition. By virtue of taking SLD as a case study, the testbed embedded an SLD within a triple representing the rule condition. Embedding an SLD within the rule condition was adopted to maintain consistency with SLD which embeds much of its information within a Rule object. Envitia used the term

hasOGCSLD to represent an OGC Filter. The property should be aligned to **style:hasOGCFilterCondition**.

A set of symbols and portrayal rules were created using this approach and then published through a SPARQL Server provided by Envitia. A separate component, the SLD Producer was provided by Geomatys to extract SLDs from the RDF documents, using information received from aviation client components. The conversion between dark and light symbols was therefore implemented within the SLD Producer. In this case, contrary to the Emergency Management Scenario there was no semantic mediation involved as the feature model was the same for both styles.

12.4 Geomatys SLD Producer WPS

The SLDProducer component has the responsibility to provide an SLD instance to the client. The client shall provide two arguments: a community ID (audience) and a Feature Type. In return, a SLD has to be returned. To allow chaining of operations, the SLDProducer had to return a raw SLD and not embedded in another response structure.



The WPS service was used to execute this creation of symbology. It is perfectly adapted as the WPS standard supports the RAWDATAOUPUT format which does not embed the result in a XML structure.

The WPS has been configured with two different processes as the two ontologies were not identical. The following sample describes the processed advertised in the GetCapabilities.

```
<wps:ProcessOfferings>
  <wps:Process wps:processVersion="1.0.0">
    <ows:Identifier> urn:ogc:cstl:wps:ows11:produceSLD </ows:Identifier>
```

```
    <ows:Title> Ows11 : ProduceSLD </ows:Title>
    <ows:Abstract> OWS11 compute SLD from Envitia Servers </ows:Abstract>
  </wps:Process>
  <wps:Process wps:processVersion="1.0.0">
    <ows:Identifier> urn:ogc:cstl:wps:ows11:RDF2SLD </ows:Identifier>
    <ows:Title> Ows11 : RDF2SLD </ows:Title>
    <ows:Abstract> OWS11 compute SLD from RDF ImageMatters Server
</ows:Abstract>
  </wps:Process>
</wps:ProcessOfferings>
```


The process description is defined as:


```
<wps:ProcessDescriptions service="WPS" version="1.0.0" xml:lang="en-EN"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:mml="http://www.w3.org/1998/Math/MathML" xmlns:ns7="http://geotoolkit.org"
xmlns:ns8="http://www.opengis.net/gml/3.2">
  <ProcessDescription storeSupported="true" statusSupported="true"
wps:processVersion="1.0.0">
    <ows:Identifier> urn:ogc:cstl:wps:ows11:RDF2SLD </ows:Identifier>
    <ows:Title> Ows11 : RDF2SLD </ows:Title>
    <ows:Abstract> OWS11 compute SLD from RDF </ows:Abstract>
    <DataInputs>
      <Input minOccurs="1" maxOccurs="1">
      <ows:Identifier> urn:ogc:cstl:wps:ows11:RDF2SLD:input:community </ows:Identifier>
      <ows:Title> Community </ows:Title>
      <ows:Abstract> Community </ows:Abstract>
      <LiteralData>
        <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string"> String
</ows:DataType>
        <ows:AnyValue/> </LiteralData>
      </Input>
      <Input minOccurs="1" maxOccurs="1">
      <ows:Identifier> urn:ogc:cstl:wps:ows11:RDF2SLD:input:typename </ows:Identifier>
      <ows:Title> Typename </ows:Title>
      <ows:Abstract> Feature type name </ows:Abstract>
      <LiteralData>
        <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string"> String
</ows:DataType>
        <ows:AnyValue/> </LiteralData>
      </Input>
    </DataInputs>
```



```
    <ProcessOutputs>
     <Output>
      <ows:Identifier> urn:ogc:cstl:wps:ows11:RDF2SLD:output:result </ows:Identifier>
      <ows:Title> Result </ows:Title>
      <ows:Abstract> Sld </ows:Abstract>
      <ComplexOutput>
       <Default>
        <Format>
         <MimeType> application/vnd.ogc.sld+xml </MimeType>
         <Encoding> utf-8 </Encoding>
         <Schema> http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd </Schema>
        </Format>
       </Default>
       <Supported>
        <Format>
         <MimeType> application/vnd.ogc.sld+xml </MimeType>
         <Encoding> utf-8 </Encoding>
         <Schema> http://schemas.opengis.net/sld/1.1.0/StyledLayerDescriptor.xsd </Schema>
        </Format>
       </Supported>
      </ComplexOutput>
     </Output>
    </ProcessOutputs>
  </ProcessDescription>
</wps:ProcessDescriptions>
```

Here an example of a call on ImageMatters Semantic Portrayal Service

```
http://ows11.geomatys.com/constellation/WS/wps/ows11?
SERVICE=WPS&
VERSION=1.0.0&
REQUEST=execute&
LANGUAGE=en-EN&
IDENTIFIER=urn:ogc:cstl:wps:ows11:RDF2SLD&
RAWDATAOUTPUT=urn:ogc:cstl:wps:ows11:RDF2SLD:output:result&
DATAINPUTS=urn:ogc:cstl:wps:ows11:RDF2SLD:input:community=h
ttp://ows.usersmarts.com/ldapp/audiences/community/Canadian
EmergencyAndDisasterManagement;
urn:ogc:cstl:wps:ows11:RDF2SLD:input:typename=ems:EMSIncide
nt
```

12.4.1 Use Case 1 : Envitia Server

For this use case, the server was supporting REST calls following patterns http://OntologyServerREST/community/typeName

The JSON –LD response contained a property called OGCSLDRule with a collection of portrayalRule, each one containing a body fragments to be included in the returned SLD with correct header.


```
{	@id: "http://1-dot-
env072015.appspot.com/resource/symbol/AviationLight/RunwayElementType",
	@type: [
		"j.1:Symbol",
		"owl:NamedIndividual",
		"foaf:Document"
	],
	audience: "http://1-dot-
env072015.appspot.com/resource/community/AviationLight",
	description: "RunwayElementType_LIGHT symbol",
	publisher: "ICAO",
	portrayalRule: [
		"http://1-dot-
env072015.appspot.com/resource/portrayalrule/AviationLight/RunwayClosedLight",
		"http://1-dot-
env072015.appspot.com/resource/portrayalrule/AviationLight/RunwayOpenLight"
	],
	label: "RunwayElementType_LIGHT",
	@context: {
		label: "http://www.w3.org/2000/01/rdf-schema#label",
		description: "http://purl.org/dc/terms/description",
		publisher: "http://purl.org/dc/terms/publisher",
		audience: {@
			id: "http://purl.org/dc/terms/audience",
			@type: "@id"
		},
		portrayalRule: {@
			id: "http://www.opengis.net/ont/portrayal/symbol#portrayalRule",
			@type: "@id"
		},
		rdfs: "http://www.w3.org/2000/01/rdf-schema#",
		geosparql: "http://www.opengis.net/ont/geosparql#",
		geo: "http://www.opengis.net/ont/geosparql#",
		foaf: "http://xmlns.com/foaf/0.1/",
		symbol: "http://www.opengis.net/ont/portrayal/symbol#",
		dct: "http://purl.org/dc/terms/",
		owl: "http://www.w3.org/2002/07/owl#",
		xsd: "http://www.w3.org/2001/XMLSchema#",
		community: "http://www.opengis.net/ont/community#",
```



```
            j.1: "http://www.opengis.net/ont/portrayal/symbol#",
            rdf: "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
            j.0: "http://purl.org/dc/terms/",
            nhd: "http://www.opengis.net/ont/testbed11/hydro/nhd#",
            skos: "http://www.w3.org/2004/02/skos/core#"
      }
}
```



```
{@
      graph: [{@
            id: "_:b0",
            @type: "j.1:OGCSLDRule",
            body: "<Rule xmlns="
            http: //www.opengis.net/se" xmlns:se="http://www.opengis.net/se"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:ogc="http://www.opengis.net/ogc" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<Name>RunwayClosedLight</Name> <ogc:Filter> <ogc:PropertyIsEqualTo>
<ogc:PropertyName xmlns:ns0="http://www.aixm.aero/schema/5.1">
ns0:timeSlice/ns0:RunwayElementTimeSlice/ns0:availability/ns0:ManoeuvringAreaAvai
lability/ns0:operationalStatus</ogc:PropertyName>
<ogc:Literal>CLOSED</ogc:Literal> </ogc:PropertyIsEqualTo> </ogc:Filter>
<MinScaleDenominator>0.0</MinScaleDenominator>
<MaxScaleDenominator>INF</MaxScaleDenominator> <PolygonSymbolizer>
<Geometry> <ogc:PropertyName
xmlns:ns0="http://www.aixm.aero/schema/5.1">ns0:timeSlice/ns0:RunwayElementTime
Slice</ogc:PropertyName> </Geometry> <Fill> <GraphicFill> <Graphic> <Mark>
<OnlineResource xlink:href="http://1-dot-
env072015.appspot.com/symbolstore/runway_closed_light.svg"/>
<Format>image/svg+xml</Format> </Mark> <Size>16</Size> </Graphic>
</GraphicFill> </Fill> <Stroke> <SvgParameter name="stroke-opacity">
<ogc:Literal>1.0</ogc:Literal> </SvgParameter> <SvgParameter name="stroke-width">
<ogc:Literal>1.0</ogc:Literal> </SvgParameter> <SvgParameter name="stroke">
<ogc:Literal>#1328BB</ogc:Literal> </SvgParameter> </Stroke>
</PolygonSymbolizer> </Rule>"
      }, {@
            id: "_:b1",
            @type: "j.1:PortrayalRuleCondition",
            hasOGCSLD: "_:b0"
      }, {@
            id: "http://1-dot-
env072015.appspot.com/resource/portrayalrule/AviationLight/RunwayClosedLight",
```

```
        @type: [
            "j.1:PortrayalRule",
            "owl:NamedIndividual"
        ],
        description: "RunwayClosedLight Rule",
        featureType: "RunwayElementType",
        hasRuleCondition: "_:b1"
    }],
    @context: {
        body: "http://www.opengis.net/ont/portrayal/symbol#body",
        description: "http://purl.org/dc/terms/description",
        featureType: "http://www.opengis.net/ont/portrayal/symbol#featureType",
        hasRuleCondition: {@
            id: "http://www.opengis.net/ont/portrayal/symbol#hasRuleCondition",
            @type: "@id"
        },
        hasOGCSLD: {@
            id: "http://www.opengis.net/ont/portrayal/symbol#hasOGCSLD",
            @type: "@id"
        },
        rdfs: "http://www.w3.org/2000/01/rdf-schema#",
        geosparql: "http://www.opengis.net/ont/geosparql#",
        geo: "http://www.opengis.net/ont/geosparql#",
        foaf: "http://xmlns.com/foaf/0.1/",
        symbol: "http://www.opengis.net/ont/portrayal/symbol#",
        dct: "http://purl.org/dc/terms/",
        owl: "http://www.w3.org/2002/07/owl#",
        xsd: "http://www.w3.org/2001/XMLSchema#",
        community: "http://www.opengis.net/ont/community#",
        j.1: "http://www.opengis.net/ont/portrayal/symbol#",
        rdf: "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
        j.0: "http://purl.org/dc/terms/",
        nhd: "http://www.opengis.net/ont/testbed11/hydro/nhd#",
        skos: "http://www.w3.org/2004/02/skos/core#"
    }
}
```


The SLD produced is the following:


```
<sld:StyledLayerDescriptor version="1.1.0" xmlns="http://www.opengis.net/se"
xmlns:se="http://www.opengis.net/se" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:sld="http://www.opengis.net/sld" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml">
  <sld:NamedLayer>
```



```
    <se:Name> RunwayElementType </se:Name>
    <sld:UserStyle>
      <FeatureTypeStyle>
        <Rule xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <Name> RunwayClosedLight </Name>
          <ogc:Filter>
            <ogc:PropertyIsEqualTo>
              <ogc:PropertyName xmlns:ns0="http://www.aixm.aero/schema/5.1">
ns0:timeSlice/ns0:RunwayElementTimeSlice/ns0:availability/ns0:ManoeuvringAreaAvai
lability/ns0:operationalStatus </ogc:PropertyName>
              <ogc:Literal> CLOSED </ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:Filter>
          <MinScaleDenominator> 0.0 </MinScaleDenominator>
          <MaxScaleDenominator> INF </MaxScaleDenominator>
          <PolygonSymbolizer>
            <Geometry>
              <ogc:PropertyName xmlns:ns0="http://www.aixm.aero/schema/5.1">
ns0:timeSlice/ns0:RunwayElementTimeSlice </ogc:PropertyName>
            </Geometry>
            <Fill>
              <GraphicFill>
                <Graphic>
                  <Mark>
                    <OnlineResource xlink:href="http://1-dot-
env072015.appspot.com/symbolstore/runway_closed_light.svg" />
                    <Format> image/svg+xml </Format>
                  </Mark>
                  <Size> 16 </Size>
                </Graphic>
              </GraphicFill>
            </Fill>
            <Stroke>
              <SvgParameter name="stroke-opacity">
                <ogc:Literal> 1.0 </ogc:Literal>
              </SvgParameter>
              <SvgParameter name="stroke-width">
                <ogc:Literal> 1.0 </ogc:Literal>
              </SvgParameter>
              <SvgParameter name="stroke">
                <ogc:Literal> #1328BB </ogc:Literal>
              </SvgParameter>
            </Stroke>
          </PolygonSymbolizer>
        </Rule>
```



```
  <Rule xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Name> RunwayOpenLight </Name>
    <ElseFilter/>
    <MinScaleDenominator> 0.0 </MinScaleDenominator>
    <MaxScaleDenominator> INF </MaxScaleDenominator>
    <PolygonSymbolizer>
     <Geometry>
      <ogc:PropertyName xmlns:ns0="http://www.aixm.aero/schema/5.1">
ns0:timeSlice/ns0:RunwayElementTimeSlice </ogc:PropertyName>
     </Geometry>
     <Fill>
      <SvgParameter name="fill-opacity">
       <ogc:Literal> 1.0 </ogc:Literal>
      </SvgParameter>
      <SvgParameter name="fill"> #888888 </SvgParameter>
     </Fill>
     <Stroke>
      <SvgParameter name="stroke-opacity">
       <ogc:Literal> 1.0 </ogc:Literal>
      </SvgParameter>
      <SvgParameter name="stroke"> #605A4E </SvgParameter>
      <SvgParameter name="stroke-width">
       <ogc:Literal> 1 </ogc:Literal>
      </SvgParameter>
     </Stroke>
    </PolygonSymbolizer>
    <TextSymbolizer>
     <Geometry>
      <ogc:PropertyName xmlns:ns0="http://www.aixm.aero/schema/5.1">
ns0:timeSlice/ns0:RunwayElementTimeSlice </ogc:PropertyName>
     </Geometry>
     <Label>
      <ogc:Function name="valueOf">
       <ogc:PropertyName xmlns:ns0="http://www.aixm.aero/schema/5.1">
ns0:timeSlice/ns0:RunwayElementTimeSlice/ns0:associatedRunway
</ogc:PropertyName>
       <ogc:Literal>
        <ogc:PropertyName xmlns:ns0="http://www.aixm.aero/schema/5.1">
ns0:timeSlice/ns0:RunwayTimeSlice/ns0:designator </ogc:PropertyName>
       </ogc:Literal>
      </ogc:Function>
     </Label>
     <Fill>
      <SvgParameter name="fill-opacity">
```

```
          <ogc:Literal> 1.0 </ogc:Literal>
        </SvgParameter>
        <SvgParameter name="fill">
          <ogc:Literal> #FFFFFF </ogc:Literal>
        </SvgParameter>
      </Fill>
    </TextSymbolizer>
   </Rule>
  </FeatureTypeStyle>
 </sld:UserStyle>
 </sld:NamedLayer>
</sld:StyledLayerDescriptor>
```

12.4.2 Use case 2 : ImageMatters Server

In use case 2, we used a SPARQL endpoint that returned a RDF payload. Using this, we could query the server with a big flexibility. Using the JENA library, we have read the RDF quite easily. The query process follows an equivalent logic, based on a CommunityID and FeatureType.

The starting point to get the rules is thus

```
PREFIX style:<http://www.opengis.net/ont/portrayal/style#>
PREFIX incident:<http://www.opengis.net/ont/emergency/incident#>
prefix dct:   <http://purl.org/dc/terms/>
PREFIX feature:<http://www.opengis.net/ont/feature#>

DESCRIBE ?rule {
  ?styleSet a style:Style;
          dct:audience
<http://ows.usersmarts.com/ldapp/audiences/community/$community>;
          style:hasRuleSet ?ruleSet.
  ?ruleSet style:hasRule ?rule.
  ?rule style:featureType ?featureType.
  ?featureType feature:gmlName 'ems:$typename'.
}
```

This request returns a set of rules for defining the symbology graphic to use for rendering and embeding in the SLD file.

With more time, we could have optimized the SPARQL query to improve performance issues encountered when a symbol has more than a dozen of rules.

12.5 WFS Sources

Due to the issues of obtaining datasets for supporting the demonstration, we decided to directly use a Linked Data representation of the Incidents. If time allowed, we could have stored the information in a WFS and then provide a semantic wrapper around WFS to convert GML to Linked Data. However, we believe it is more optimal to provide a service that access directly an incident database and returns directly the information as Linked Data. Given that Semantic Mediation Service performs the mediation of information on Linked Data representation, it makes sense to get this information available directly through standard RESTful Linked Data API.

12.6 FPS and Client

FCU provided a map client interacting with the semantic portrayal service and the FPS. The original scenario in FCU implementation plan used the following steps:

1. User selects a bounding box on the map.
2. Client resolves the response from SPARQL endpoint, and let user chooses a symbol domain.
3. Client sends request to FPS with a bounding box and symbol domain to FPS.
4. FPS response is sent to the client.
5. Client reveals the result on the map.

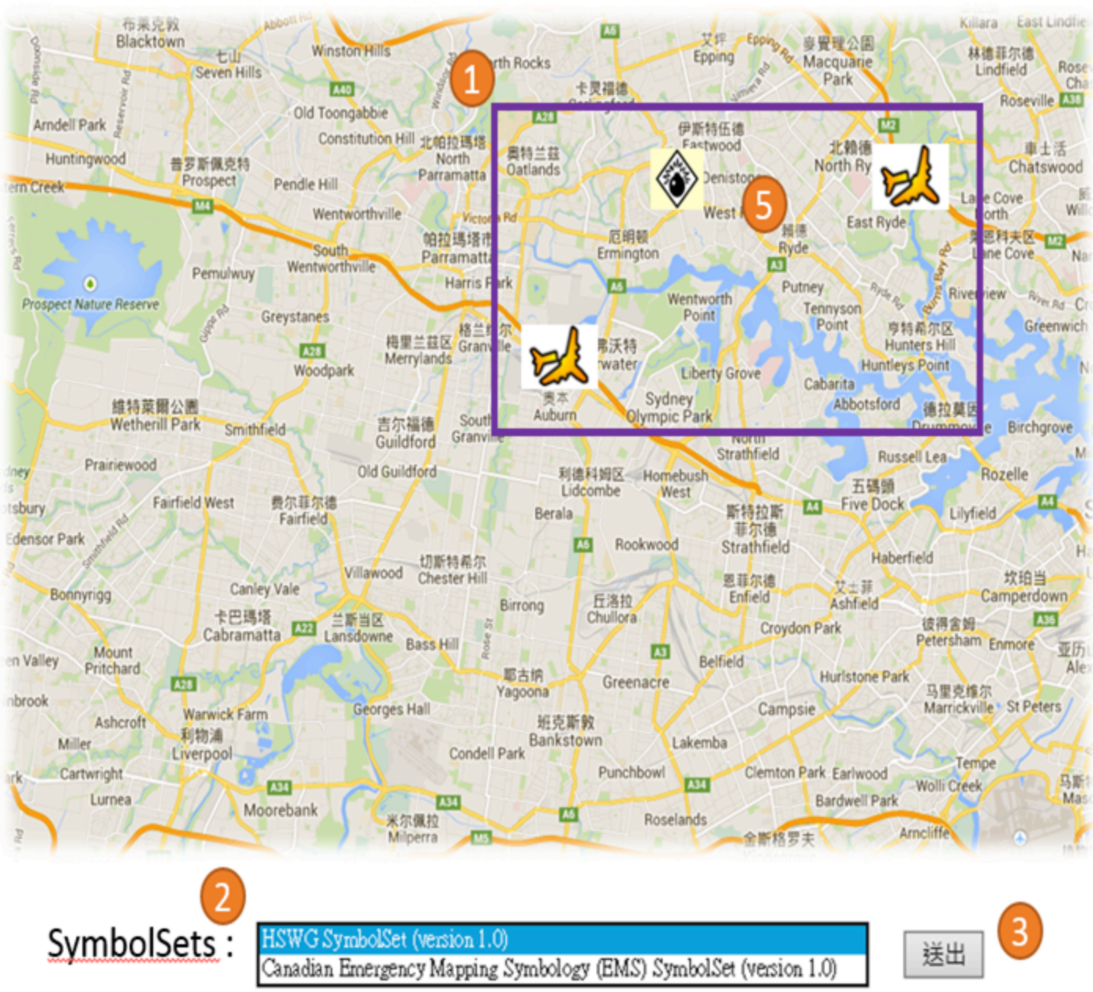


Figure 14 FCU Map Client

FCU did implement a function to get symbol sets in the first iteration, with two symbol types (see Figure 14 FCU Map Client).

In the first iteration, a client would get symbol sets from the SPARQL endpoint directly from the semantic portrayal service then send selected symbol sets and a bounding box as parameters to FPS. A FPS, which is an OGC WMS, can generate comparative picture and return it to a client.

In the latest version of the sequence event flow, the client provided a CommunityURI and FeatureType to FPS. The CommunityURI would get from Semantic Portrayal Server, and FeatureType will get froma FPS. From the perspective of implementation, this is considered as unnecessary since both CommunityURI and FeatureType can be returned from a FPS instance. This would simplify the complexity of the client and improve performance.

However, FPS didn't implement the emergency symbol set due to lack of time, but the final sequence may look like the one illustrated in Figure 15 FCU Sequence Diagram.

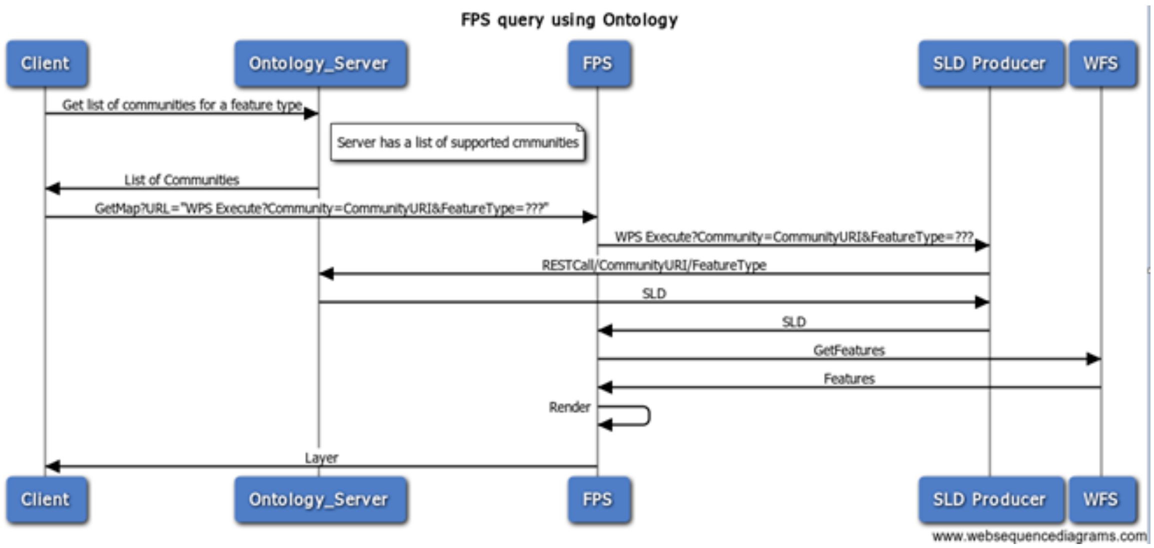


Figure 15 FCU Sequence Diagram

This task will need to be further investigated in the future and the implications with using FPS with Linked Data as an alternative format to get feature information other than GML.

13 Challenges encountered

The most difficult task of the semantic symbology mediation was to solve the semantic mediation challenge by leveraging existing Linked Data standards. The proposed SPARQL Extension and Semantic Mediation ontologies provide a solid framework to define semantic mapping in an extensible way. The SPARQL extension ontology has a powerful metamodeling framework allowing creating new types of mapping and functions. The framework will need to be further tested on more use cases.

Another challenge encountered during this testbed was to provide service APIs that are simple enough to implement and reproduce using mainstream tools but also easy to integrate with existing client technologies. The choice of a RESTful API proved that the integration of the new services can be performed very quickly. The use of Linked Data representation is adequate for machine processing. However, more work needs to be done on the JSON-LD serialization to make it more friendly and consumable by web clients.

The biggest challenge is the tension existing between Linked Data APIs and the current OGC Service Oriented APIs such as WFS and FPS which provide only syntactic encoding based on GML. There is not clear path as to how to convert GML to RDF model in a systematic way. To perform the semantic mediation and the symbology portrayal, the implementation is significantly simplified when Linked Data is used all the way through the process. More investigations needs to be done in the future how GML and Linked Data representation can be reconciled without creating too overhead to convert one representation to another.

At last, the lack of good datasets during the testbed to exercise the demo scenario has impacted the implementation, integration and testing of the different components of the architecture, preventing us from accomplishing the whole workflow to display incidents with the mediated symbology on the client.

14 Recommendation for future work

The results of the testbed have been very fruitful and a number of breakthroughs were achieved. For the first time we have a formal model to represent semantic mediation mappings and the ability to extend SPARQL endpoints with new capabilities (sharable functions and rules based on SPARQL). Second we have a solid foundation to represent portrayal information semantically and thus making them more sharable and machine-processable by different services. Third, the use of RESTful services demonstrated that they are a good fit for Linked Data, the use of JSON-LD and web clients.

We strongly recommend that these breakthroughs are leveraged in the next testbed to bring them to a level of maturity and robustness to become future standards. The following areas need to be investigated:

Semantic Portrayal Service

The REST API for the Semantic Portrayal Service needs to be completed by providing endpoints to access and create, update and delete styles, rules, graphics information.

The Portrayal Ontologies needs to be completed by formalizing further the Graphics ontology by defining graphic objects and attributes for lines and areas. The Symbology ontology needs to be refined further to accommodate line and area-based symbols and well as composition of multiple symbols and their bindings with the geometric properties of features. The Portrayal Catalog ontology needs to be refined to get a solid model for managing registry of styles. We should also consider extending the Semantic Portrayal Service by providing a rendering endpoint to convert a Linked Data Model to a symbolic representation in well-known formats such as SVG or KML.

Semantic Mediation

The SPARQL Extension ontology needs to be further refined and documented and then exercised on a variety of use cases to reach a level of maturity and robustness needed to become a standard. More implementations leveraging this ontology should be pursued to validate the feasibility of using this standard to extend SPARQL endpoint capabilities.

The REST API for Semantic Mediation Service needs to be tested further and the serialization in JSON-LD needs to be improved to lower the bar of integration with web clients. Other use cases for the use the SMS needs to be investigated, such as query rewriting service (a SPARQL query for one source ontology to be converted to one or more SPARQL queries for the target ontology).

Datasets

The need of good datasets to support demo scenarios at the start of next testbed is crucial to get an effective execution of the testbed.

Annex A

Portrayal Ontologies

The documentation of the portrayal ontologies is available at the following endpoints:

Portrayal Style ontology

http://ows.usersmarts.com/owldocgen/owldoc?url=http://www.opengis.net/ont/portrayal/style

Symbology ontology

http://ows.usersmarts.com/owldocgen/owldoc?url=http://www.opengis.net/ont/portrayal/symbol

Graphic Ontology

http://ows.usersmarts.com/owldocgen/owldoc?url=http://www.opengis.net/ont/portrayal/graphic

Portrayal Catalog

http://ows.usersmarts.com/owldocgen/owldoc?url=http://www.opengis.net/ont/portrayal

Annex B

Semantic Mediation Ontologies

The SPARQL extensions ontology is available at:

http://ows.usersmarts.com/owldocgen/owldoc?url=http://www.opengis.net/testbed11/ont/geosparql/extensions

The Mediation ontology is available at:

http://ows.usersmarts.com/owldocgen/owldoc?url=http://www.opengis.net/testbed11/ont/geosparql/mediation

Revision history

Date	Release	Editor	Primary clauses modified	Description
2015-06-07	0.1	Stephane Fellah	Outline of document	
2015-06-14	0.2	Stephane Fellah	Portrayal ontologies section	Portrayal ontologies documentation
2015-06-24	0.3	Stephane Fellah	SPARQL extensions ontologies section	SPARQL extension ontology documentation
2015-07-05	0.4	Stephane Fellah	Section 1-11	Finalizing section 1-11
2015-07-06	1.0	Stephane Fellah	Section 11-14+annexes	Finalizing remaining sections
2015-07-20	1.0r1	Stephane Fellah	All	Add contribution from FCU (Section 12.6) and update from feedback on previous versions
2015-10-09		Carl Reed	Various	Prepare for publication
2015-05-01		Scott Simmons	All	Finalize for publication

Bibliography

[1] Guidelines for Successful OGC Interface Standards, OGC document 00-014r1

[2] SKOS Reference, W3C Recommendation, 18 August 2009. Latest version available at [http://www.w3.org/TR/skos-reference .](http://www.w3.org/TR/skos-reference)

[3] RDF 1.1 Turtle: Terse RDF Triple Language. W3C Recommendation, 25 February 2014. The latest edition is available at http://www.w3.org/TR/turtle/

[4] Gruber, Thomas R. "Toward principles for the design of ontologies used for knowledge sharing?" International journal of human-computer studies 43, no. 5 (1995): 907-928.

[5] Stuckenschmidt, Heiner, and Michel Klein. "Structure-based partitioning of large concept hierarchies." In The Semantic Web–ISWC 2004, pp. 289-303. Springer Berlin Heidelberg, 2004.

[6] Stuckenschmidt, Heiner, Christine Parent, and Stefano Spaccapietra, eds.Modular ontologies: concepts, theories and techniques for knowledge modularization. Vol. 5445. Springer, 2009.

