

Testbed-12 Semantic Portrayal,  
Registry and Mediation Engineering  
Report

# Table of Contents

1. Introduction .....	5
1.1. Scope .....	5
1.2. Document contributor contact points .....	5
1.3. Future Work .....	6
1.3.1. SRIM and ISO 19115 mapping .....	6
1.3.2. SRIM Layer and Map Profile .....	6
1.3.3. Pubsub and federation of Registry .....	6
1.3.4. Web of Vocabulary Ontology and Service .....	6
1.3.5. Application of Shape Constraint Language (SHACL) for Linked Data .....	8
1.3.6. Composite Symbology and alternates renderers for Semantic Portrayal Service.....	8
1.4. Foreword .....	9
2. References .....	10
3. Terms and definitions.....	12
3.1. feature.....	12
3.2. interoperability .....	12
3.3. map .....	12
3.4. model .....	12
3.5. ontology .....	12
3.6. portrayal.....	12
3.7. semantic interoperability.....	12
3.8. semantic mediation .....	13
3.9. symbol.....	13
3.10. symbology encoding .....	13
3.11. syntactic interoperability.....	13
4. Conventions .....	14
4.1. Abbreviated terms .....	14
5. Overview .....	16
6. Status Quo & New Requirements Statement .....	17
6.1. Status Quo .....	17
6.1.1. Semantic Registry Service .....	17
6.1.2. Semantic Mediation Service .....	18
6.1.3. Semantic Portrayal Service .....	18
6.2. Requirements Statement .....	19
6.2.1. Semantic Registry Service .....	19
6.2.2. Semantic Mediation Service .....	20
6.2.3. Semantic Portrayal Service .....	20
7. Solutions .....	21
7.1. Targeted Solutions .....	21

7.1.1. Semantic Registry Service Targeted Solutions .....	21
7.1.2. Semantic Mediation Service Targeted Solutions.....	22
7.1.3. Semantic Portrayal Service Targeted Solutions .....	23
7.2. Recommendations .....	23
7.2.1. Semantic Registry Service Recommendations .....	23
7.2.2. Semantic Mediation Service Recommendations.....	24
7.2.3. Semantic Portrayal Service Recommendations .....	24
8. Semantic Registry Service .....	26
8.1. Overview .....	26
8.2. Review of existing standards .....	27
8.2.1. DCAT .....	27
8.2.2. DCAT-AP .....	28
8.2.3. GeoDCAT-AP .....	28
8.2.4. Asset Description Metadata Schema (ADMS).....	29
8.2.5. Project Open Data (POD).....	30
8.2.6. ISO 19115-1.....	30
8.2.7. ISO 19135 .....	31
8.2.8. Shapes Constraint Language (SHACL).....	31
8.3. Semantic Registry Information Model (SRIM) .....	32
8.4. Implementations .....	32
8.4.1. Semantic Mapping to SRIM .....	32
8.4.2. ISO 19139 Mapping Issues.....	33
8.4.3. Semantic Registry Service .....	42
8.4.4. Semantic Registry Service REST API .....	44
8.4.5. Integration with OGC Catalog Services .....	45
8.4.6. Integration with Clients .....	52
9. Semantic Mediation Service .....	57
9.1. Overview .....	57
9.2. Analysis .....	57
9.3. SRIM Schema Application Profile.....	58
9.4. Schema Registry CSW-ebRIM profile.....	66
9.5. Implementations .....	67
9.5.1. Integration of Galdos Schema Registry with Semantic Registry.....	67
9.5.2. Semantic Mediation Service .....	71
10. Semantic Portrayal Service .....	73
10.1. Overview .....	73
10.2. Review of symbology graphic formats .....	73
10.2.1. Symbol Encoding (SE) .....	73
10.2.2. OGC Style Layer Descriptors (SLD).....	73
10.2.3. Scalable Vector Graphics (SVG).....	73
10.2.4. ISO 19117 .....	74

10.2.5. KML .....	74
10.3. Portrayal Ontologies .....	74
10.3.1. Style Microtheory .....	75
10.3.2. Symbol Microtheory .....	75
10.3.3. Symbolizer Ontology .....	76
Appendix A: Semantic Registry Information Model (SRIM) .....	81
Namespaces .....	81
Registry Core Ontology .....	82
Overview .....	82
srim:Register .....	85
srim:RegisterEntry .....	87
srim:ItemClass .....	89
srim:Item .....	90
prov:Activity .....	94
vcard:Address .....	94
foaf:Agent .....	95
prov:Attribution .....	95
skos:Concept .....	96
skos:ConceptScheme .....	97
foaf:Document .....	97
vcard:Email .....	98
extent:GeographicExtent .....	98
id:Identifier .....	99
dct:LicenseDocument .....	100
link:Link .....	100
dct:Location .....	101
gr:OpeningHoursSpecification .....	101
org:Organizaton .....	102
dct:PeriodOfTime .....	102
vcard:Phone .....	103
foaf:Project .....	103
dct:ProvenanceStatement .....	104
srim:Release .....	104
dct:RightsStatement .....	105
dct:Standard .....	106
vcard:VCard .....	107
Dataset Application Profile .....	108
dcat:Dataset .....	108
dcat:Distribution .....	108
dct:FileFormat .....	109
Service Application Profile .....	110

srin:Service .....	110
srin:APIDocument .....	110
Appendix B: SRIM Schema Application Profile .....	112
Semantic Mediation Ontology .....	112
Overview .....	112
schema:Schema .....	113
schema:SchemaMapping .....	116
schema:SchemaLanguage .....	117
schema:SchemaMappingLanguage .....	118
schema:SchemaRelease .....	118
Appendix C: Semantic Portrayal Ontologies .....	120
Style Ontology .....	121
Style .....	121
FeatureTypeStyle .....	122
CoverageStyle .....	123
PortrayalRuleSet .....	123
PortrayalRule .....	124
RuleCondition .....	125
PortrayalRuleList and RuleItem .....	128
Symbology Ontology .....	129
SymbolSet .....	130
Symbol .....	131
Symbolizer Microtheory .....	133
Symbolizer .....	134
Point Symbolizer .....	134
Line Symbolizer .....	134
Polygon Symbolizer .....	135
Text Symbolizer .....	135
Raster Symbolizer .....	136
Composite Symbolizer .....	136
Graphics Microtheory .....	136
Context .....	136
Scope .....	137
Terminology used in the Graphics Ontology .....	137
Graphics Ontology Classes .....	137
Graphic Datatypes .....	142
Graphic Properties .....	143
Appendix D: Semantic Registry Service REST API v0.1 .....	151
Overview .....	151
HTTP Verbs .....	151
HTTP Status Codes .....	152

Headers .....	152
Errors .....	152
Paging and Sorting .....	153
Search Results .....	155
Resources Summary .....	157
Link Relation Types .....	158
Level 2 REST Endpoints .....	160
Content Negotiation .....	163
Semantic Registry Resources .....	164
Service Root .....	164
Capabilities .....	167
JSON-LD Context .....	170
Registers .....	178
Register JSON Schema .....	178
Register .....	186
Items .....	193
Item .....	204
SPARQL Service .....	210
Harvester Service Resources .....	212
Service Root .....	212
Harvester Types .....	214
Harvester Type .....	219
Harvester Sources .....	221
Harvester Source .....	227
Harvest Action Resource .....	233
Appendix E: Semantic Mediation Service REST API .....	237
Overview .....	237
HTTP verbs .....	237
HTTP status codes .....	238
Headers .....	238
Errors .....	239
Paging and Sorting .....	239
Search Results .....	241
Resources Summary .....	243
Link relation types .....	244
Content negotiation .....	245
Resources descriptions .....	245
Service Entry Point .....	245
Capabilities .....	248
JSON-LD Context .....	251
Schema Collection .....	253

JSON Schema .....	253
Schema .....	258
Schema Mapping Collection .....	263
JSON Schema .....	263
Schema Mapping .....	266
Validator .....	271
Transformer Resource .....	276
Appendix F: Semantic Portrayal Service REST API .....	277
Overview .....	277
HTTP verbs .....	277
HTTP status codes .....	277
Headers .....	278
Errors .....	278
Paging and Sorting .....	279
Search Results .....	281
Resources Summary .....	283
Level 2 REST Endpoints .....	284
Link relation types .....	287
Content negotiation .....	288
Semantic Portrayal Resources .....	289
Service Root .....	289
Capabilities .....	292
JSON-LD Context .....	294
Styles .....	301
Style .....	306
Render Geospatial Data (Proposed) .....	311
Symbols .....	311
Symbol .....	316
Symbol Renderer .....	322
SymbolSets .....	324
SymbolSet .....	329
SPARQL Service .....	334
Appendix G: Revision History .....	338
Appendix H: Bibliography .....	339

Publication Date: 2017-04-25

Approval Date: 2016-03-09

Posted Date: 2016-12-02

Reference number of this document: OGC 16-059

Reference URL for this document: <http://www.opengis.net/doc/PER/t12-A066>

Category: Public Engineering Report

Editor: Stephane Fellah

Title: Testbed-12 Semantic Portrayal, Registry and Mediation Engineering Report

---

## **OGC Engineering Report**

### **COPYRIGHT**

Copyright © 2017 Open Geospatial Consortium. To obtain additional rights of use, visit <http://www.opengeospatial.org/>

### **WARNING**

This document is an OGC Public Engineering Report created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard and 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. However, the discussions in this document could very well lead to the definition of an OGC Standard.



## LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by

destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

## **Abstract**

This engineering report documents the findings of the activities related to the Semantic Portrayal, Registry and Mediation components implemented during the OGC Testbed 12. This effort is a continuation of efforts initiated in the OGC Testbed 11. This report provides an analysis of the different standards considered during this effort, documents the rendering endpoints extension added to the Semantic Portrayal Service and the migration of the Portrayal metadata to the Semantic Registry, which is aligned with the DCAT REST Service API. We also discuss the integration of the CSW ebRIM for Application Schema with the Semantic Mediation Service, and document the improvements of the SPARQL Extensions, Portrayal and Semantic Mediation ontologies defined in the previous testbed.

## **Business Value**

Catalog services usually provide discovery of data and services; however, the ability to discover other related resources that can help applications better understand and render the data are not commonly found. For example, getting available styles for a layer or a feature type. In Testbed 12 it was advanced the use of W3C semantic technologies to better integrate datasets, services, schemas, schema mappings, portrayal information, and layers.

## **What does this ER mean for the Working Group and OGC in general**

This engineering report is important to the OGC Geosemantics Domain Working Group as it advances the semantic enablement of geospatial information found in catalogs such dataset, service and portrayal metadata potentially providing a bridge between the geospatial and semantic web communities. The testbed also produced a number of ontologies for portrayal, schema management and registry.

## **Keywords**

ogcdocs, testbed-12, CSW, eb-RIM, catalogue, metadata, SPARQL, RDF, OWL, ontology, semantic web, linked data, DCAT, Portrayal, Schema, Schema Mapping, Hypermedia, REST.

## **Proposed OGC Working Group for Review and Approval**

This engineering report will be submitted to the Geosemantics Domain Working Group for review.

# Chapter 1. Introduction

Catalog services usually provide discovery of data and services; however, the ability to discover other related resources that can help applications better understand and render the data are not commonly found. For example, getting available styles for a layer or a feature type. This report captures the work performed in Testbed 12 in setting up three types of semantic services based on W3C semantic technologies to better integrate datasets, services, schemas, schema mappings, portrayal information, and layers.

1. The **Semantic Registry Service** allows discovering and search of geospatial assets (e.g. datasets, services, schemas, portrayal information, and layers). It is based on: the W3C Data Catalog Vocabulary (DCAT); the W3C PAV - Provenance, Authoring and Versioning ontology; and the Dublin Core Metadata Terms. The Service is based on a new developed ontology called Semantic Registry Information Model (SRIM) which generalizes the DCAT model to accommodate other types of geospatial assets. The service API is an hypermedia-driven and uses JSON-LD, Linked Data and Hypermedia Application Language (HAL).
2. The **Semantic Mediation Service** provides the ability to perform transformation of data from one schema to another, including chaining of transformation. It is based on a SRIM application profile for describing schemas and schema mappings. The service is an hypermedia-driven REST API.
3. The **Semantic Portrayal Service** provides the ability to render datasets to multiple output formats (e.g. SVG and PNG) and generate different styling encoding (e.g. SLD, MapCSS, and CartoCSS). It is based a set of portrayal ontologies for styles, symbols and graphics. It uses a SRIM profile to represent portrayal metadata.

## 1.1. Scope

This OGC document specifies semantic information models and REST APIs for Semantic Registry, Semantic Mediation and Semantic Portrayal Services. It introduces the Semantic Registry Information Model (SRIM), a *superset* of the W3C DCAT ontology. SRIM can accommodate registry items other than `dcat:Dataset`, such as Service description, Schema and Schema Mapping description, and Portrayal Information (Styles, Portrayal Rules, Graphics and Portrayal Catalog), Layer, and Map Context. The Semantic Registry Service is used as an integration solution for federating and unifying information produced by different OGC Catalog Service information by providing a simplified access through hypermedia-driven API, using JSON-LD, Linked Data and HAL-JSON. During the testbed, the Semantic Registry was used to store information about geospatial datasets and services, schemas and portrayal information. The Semantic Mediation Service was used to validate and perform transformation between schemas, including transformation chaining. The Semantic Portrayal Service was used as a convenience API to access Portrayal Registry information and perform rendering of geospatial data to different graphic representation (SVG, Raster and other pluggable formats).

## 1.2. Document contributor contact points

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

Table 1. Contacts

Name	Organization
Stephane Fellah	Image Matters LLC
Gobe Hobona	Envitia
Richard Martell	Galdos Inc
Luis Bermudez	OGC

## 1.3. Future Work

### 1.3.1. SRIM and ISO 19115 mapping

The SRIM ontology in Testbed 12 used Dublin Core Metadata Terms. Future work to enhance SRIM is to map it with ISO 19115. Request for changes to improve the current standard ISO 19115 might also be required to better align with Linked Data. ISO 19115 should provide better use of controlled vocabularies, linked data friendly identifiers, and a better service description that enables automated access to services.

### 1.3.2. SRIM Layer and Map Profile

There is no standard way to describe metadata for layers and maps. While layers and maps are derived from a Dataset, they have their own specific metadata. Future work could investigate a profile description for layers and maps that extends the Registry Item and relates them to Datasets, Services and Portrayal Information. The description will be used by the Semantic Registry and Semantic Portrayal Service.

### 1.3.3. Pubsub and federation of Registry

In Testbed 12 , the Semantic Registry harvested information from a federation of CSW services to exercise the Semantic Registry Information Model (RIM) and the REST API. Future work could include improving the efficiency of the harvesting process by investigating the publish/subscribe protocol and versioning management of the register items in the Semantic Registry as they change over time.

### 1.3.4. Web of Vocabulary Ontology and Service

With the deployment of the Semantic Web and Linked Open Data, data sources have multiplied, as well as, the machine-processable **controlled vocabularies** that structure and constrain the interpretations of these data. These controlled vocabularies can be ontologies (RDF Schema, OWL), codelists, taxonomies, thesauri (SKOS) sometimes augmented with additional rules (SPARQL/SPIN rules, SWRL, RIF) ,and constraints (SHACL).

Vocabulary directories exist (e.g. LOV), but there is an ever-increasing demand for environments that simplify searching, editing and collaborative contributions to the vocabularies by non-experts of the Semantic Web. This creates a tension between very rich formalisms and a need to

democratize participation in the life cycle of controlled vocabularies.

Vocabularies are most likely to be adopted and shared if they are made available easily. Nevertheless, despite successes in the use of SKOS for encoding vocabularies, current standards provide only low-level interfaces to vocabulary data. For example, many vocabularies are published as an RDF document for download. However, if the vocabulary is large, then the download will be commensurately large; if the user only wants to retrieve a single vocabulary term or select a few terms, this option requires processing on the client side. Alternatively, access to vocabularies is often provided at a SPARQL endpoint. SPARQL is the generic RDF query language. While it is powerful, it is also considered a low-level language similar to the relational database query language SQL and normally is only used by database administrators.

Some SKOS vocabularies are published via other HTTP interfaces. However, each implementation uses different protocols and supports a varied set of features (e.g. content-negotiation provided by the GEMET REST interface and NERC Data Grid's Vocabulary Server SOAP interface). In some cases, one or both of human-readable formats and machine-readable formats are not available. Thus, discovery and access across vocabulary endpoints becomes challenging and ad-hoc.

There is a clear opportunity to design an API to match the SKOS and OWL vocabularies, taking advantage of the fact that most modern vocabulary content is structured using SKOS and OWL classes and predicates. This API can then be used as the basis for various higher-level vocabulary applications (NLP applications, Concept Recommender, Semantic Enricher, etc.) that can be used to enrich for example ISO 19115 metadata and other OGC services using controlled vocabularies in their metadata.

The Testbed 11 and 12 had explored the high-level description of ontologies and schemas to support semantic mediation. Future work can include investigation of the kind of metadata needed to enable search on controlled vocabularies by defining an ontology that addresses the following aspects:

- Classification of Vocabulary types
- Relationships to other vocabularies (extensions, imports, specialization, metadata vocabularies, etc).
- Statistical information about vocabularies (number of concepts, concept schemes, classes, properties, instances, datatypes)
- Schema encoding (OWL, RDF Schema, SKOS)
- Expressiveness
- Preferred prefix
- Preferred Namespace uri
- Governance metadata
- Versioning information

Based on this ontology, we propose to define a standard REST API to search and access vocabulary metadata and their terms using best practices in REST API (hypermedia driven for example).

### 1.3.5. Application of Shape Constraint Language (SHACL) for Linked Data

The Semantic Registry Information Model (SRIM), developed during Testbed 12, is defined as a superset of W3C DCAT standard and encoded as an OWL ontology. However, the OWL does not capture some of the semantic integrity constraints that are necessary to validate the instance information encoded using the SRIM ontology profiles. This is not an isolated problem. The DCAT ontology, for example, defines a set of classes and properties and reuses a large number of external vocabularies such as Dublin Core, but does not provide any restrictions in the ontology. Users have to read the profile documents such as DCAT-AP or GeoDCAT-AP to know which and how properties should be applied for a given class (mandatory, recommended or optional). For example, a Dataset could have only one title per language, or contact information should have either a person name, organization name or position name, and either email or telephone number. These kind of restrictions cannot be captured with OWL, and until now it required human interpretation to implement the constraints in code.

To fill these gaps, the emerging W3C standard called [Shape Constraint Language \(SHACL\)](#) provides a powerful framework to define the "shape" of the graph data and the ability to define complex integrity constraints using well-defined constraints constructs defined in RDF and SPARQL/Javascript constraints. SHACL is not a replacement of RDFS/OWL, but a complementary technology that is not only very expressive but also highly extensible. While RDFS and OWL are used to define vocabularies terms (classes/properties) and their hierarchies (subclasses, subproperties), as well as the nature of the classes and properties (union, intersection, complement of classes, transitive, inverse, symmetric properties, etc.), SHACL is more appropriate to capture the property constraints (cardinality, valid values or shape values and interdependencies between them) and capable of accommodating multiple profiles by providing different shapes for the same ontology. The SHACL vocabulary is not only defined in RDF itself, but the same macro mechanisms can be used by anyone to define new high-level language elements and publish them on the web. This means that SHACL will not only lead to the reuse of data schemas but also to domain-specific constraint languages. Furthermore, SHACL can be used in conjunction with a variety of languages beside SPARQL, including JavaScript. Complex validation constraints can be expressed in JavaScript so that they can be evaluated client-side. In addition, SHACL can be used to generate validation reports for quality control with potentially suggestions to fix validation errors. Overall, **SHACL is a future-proof schema language designed for the Web of Data.** While SHACL is not yet a standard, there are already existing implementations using it (e.g. Topbraid).

Future work could include investigation of the use of SHACL shapes to define application profiles, generation and data entry, data validation, and quality control of linked data information.

### 1.3.6. Composite Symbology and alternates renderers for Semantic Portrayal Service.

During the Testbed 11, it was introduced the portrayal ontology that focused on point-based symbology (icons for Emergency Management). Testbed 12 extended this work by providing a richer symbolizer and graphics ontology that can accommodate line and area-based symbols along with graphic attributes applicable to these symbols. Future work could extend the ontology to accommodate more complex symbology including composite symbols and symbol templates. The extended ontology will help describe more advanced symbology standards such as the family of MIL2525 symbols.

This Testbed managed to render symbol legend based on their definition, however more work is needed to develop rendering a SVG map based on the portrayal ontology. Future work can lay out the foundation to express styles that have at least the same expressiveness as SLDs. The proposed work can extend the portrayal ontology to represent composite symbols and symbol templates. Related future work can include investigation of other renderer outputs such as JSON encoding of the portrayal information, so they can be handled on the client side in HTML5 Canvas or other rendering libraries such as D3.js. Other renderers may also investigate SLD production from the RDF descriptors and investigate how unsupported features from the portrayal ontology can be supported in less expressive graphic languages than SVG, such as KML.

## **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.



# Chapter 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 16-062 - OGC® Testbed-12 Catalogue and SPARQL Engineering Report
- OGC 15-058 - OGC® Testbed-11 Symbology Mediation Engineering Report
- OGC 15-054 - OGC® Testbed-11 Implementing Linked Data and Semantically Enabling OGC Services Engineering Report
- OGC 13-084r2, OGC I15 (ISO19115 Metadata) Extension Package of CS-W ebRIM Profile .0, 2014
- OGC 12-168r6, OGC® Catalogue Services 3.0 - General Model, 2016
- OGC 11-052r4, OGC GeoSPARQL- A Geographic Query Language for RDF Data, 2011
- OGC 08-125r1, KML Standard Development Practices, Version 0.6, 2009.
- OGC 07-147r2, KML Version 2.2.0.2008
- OGC 07-110r4, CSW-ebRIM Registry Service ebRIM profile of CSW (.0.1), 2009
- OGC 07-045, OGC Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile (.0.0), 2007
- OGC 07-006r1, OpenGIS Catalogue Service Implementation Specification 2.0.2, 2007
- OGC 06-129r1, FGDC CSDGM Application Profile for CSW 2.0 (0.0.12), 2006
- OGC 06-121r9, OGC® Web Services Common Standard
- OGC 06-121r3, OpenGIS® Web Services Common Specification, version 1.1.0 with Corrigendum 1 2006
- OGC 05-078r4, OpenGIS Styled Layer Descriptor Profile of the Web Map Service Implementation Specification, Version 1.1.0, 2006
- OGC 05-077r4, OpenGIS® Symbology Encoding Implementation Specification, Version 1.1.0, 2006.
- ISO/TS 19139:2007, Geographic information — Metadata — XML schema implementation
- ISO 19119:2005, Geographic information — Services
- ISO 19117:2012, Geographic information — Portrayal
- ISO 19115:2003, Geographic information — Metadata
- ISO 19115:2003/Cor 1:2006, Geographic information — Metadata
- ISO 19115-1:2014, Geographic information — Metadata — Part 1: Fundamentals
- Dublin Core Metadata Initiative, last visited 12-09-2016, available from <http://dublincore.org/>
- NSG Metadata Foundation (NMF) – Part 1: Core, version 2.2, 23 September 2014 <https://nsgreg.nga.mil/doc/view?i=4123>
- DGIWG 114, DGIWG Metadata Foundation (DMF),last visited 12-09-2016, available from [https://portal.dgiwg.org/files/?artifact\\_id=9189&format=pdf](https://portal.dgiwg.org/files/?artifact_id=9189&format=pdf)

- DoD Discovery Metadata Specification (DDMS),last visited 12-09-2016, available from <https://metadata.ces.mil/dse-help/DDMS/index.htm>
- SPARQL Protocol and RDF Query Language (SPARQL),last visited 12-09-2016, available from <https://www.w3.org/TR/rdf-sparql-query>
- DCAT, last visited 12-09-2016, available from <https://www.w3.org/TR/vocab-dcat/>
- National System for Geospatial Intelligence Metadata Implementation Specification (NMIS) – Part 2: XML Exchange Schema
- Project Open Data Metadata Schema v1.1 <https://project-open-data.cio.gov/v1.1/schema/>
- Asset Description Metadata Schema (ADMS) <https://www.w3.org/TR/vocab-adms/>
- JSON-LD 1.0 <https://www.w3.org/TR/json-ld/>

# Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9] and in OGC® Abstract Specification Topic TBD: TBD 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

# Chapter 4. Conventions

## 4.1. Abbreviated terms

- API Application Program Interface
- CRS Coordinate Reference System
- CSW Catalog Services for the Web
- DCAT Data Catalog Vocabulary
- DCAT-AP DCAT Application Profile for Data Portals in Europe
- DCMI Dublin Core Metadata Initiative
- EARL Evaluation and Report Language EU European Union
- EuroVoc Multilingual Thesaurus of the European Union
- GEMET GEneral Multilingual Environmental Thesaurus
- GML Geography Markup Language
- GeoDCAT-AP Geographical extension of DCAT-AP
- IANA Internet Assigned Numbers Authority
- INSPIRE Infrastructure for Spatial Information in the European Community
- ISO International Standardisation Organisation
- JRC European Commission - Joint Research Centre MDR Metadata Registry
- N3 Notation 3 format
- NAL Named Authority Lists
- OGC Open Geospatial Consortium
- OWL Web Ontology Language
- RDF Resource Description Framework
- RFC Request for Comments
- SE Symbology Encoding
- SLD Style Layer Descriptor
- SKOS Simple Knowledge Organization System
- SPARQL SPARQL Protocol and RDF Query URI Uniform Resource Identifier
- SVGScalable Vector Graphics
- TTL Turtle Format
- URIUnique Resource Identifier
- URLUniform Resource Locator
- URNUniform Resource Name
- W3C World Wide Web Consortium

- WG Working Group
- WKT Well Known Text
- XML eXtensible Markup Language
- XSLT eXtensible Stylesheet Language Transformations

# Chapter 5. Overview

This engineering report includes the following major sections:

- **Status Quo & New Requirements Statement:** This section describes current standards and requirements for developing Semantic Registry, Semantic Mediation and Semantic Portrayal Services.
- **Solutions:** This section describes the solution architectures considered by the testbed, as well as the solution architecture implemented by the testbed. We organized the reporting of the activities of this thread around the services implemented.
- **Appendix sections:** These sections include semantic models and REST API of each service.

# Chapter 6. Status Quo & New Requirements Statement

## 6.1. Status Quo

### 6.1.1. Semantic Registry Service

Current Catalog solutions are highly dependent upon the metadata model employed for the service and data descriptions. Many of today's service instances and data holdings are based on an ISO 19115 metadata model using XML as the exchange format. These solutions are very document-centric due to the nature of XML Document Object Model (DOM) that are used to populate these catalogs. This is often a challenge when information need to be integrated and be linked together, as it often requires to use complex protocols (XLink, CSW GetRecords operations) to circumvent the lack of global unique identifier system in XML document. This often leads to significant overhead to query across multiple documents and difficulties in reusing and linking information.

Linked Data provides a solution as information are modeled as a Directed Labeled Graph. The nodes of the graph (called **resources**) and edges (called **properties**) are identified with globally unique resource identifiers (URIs) which are assigned to a unique meaning (defined in **ontologies**). While best practices require to have resolvable resource URIs in Linked Data, in practice it is often not the case. Linked Data from different namespaces are usually aggregated in catalogs to facilitate search and discovery and get easy access to these resource descriptions.

The W3C has released the DCAT recommendation in early 2014. "DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. By using DCAT to describe datasets in data catalogs, publishers increase discoverability and enable applications easily to consume metadata from multiple catalogs. It further enables decentralized publishing of catalogs and facilitates federated dataset search across sites." (W3C). Thus, DCAT defines a standard way to publish machine-readable metadata about a dataset. It does not make any assumptions about the format of the datasets described in a catalog.

Since then, a number of profiles of DCAT have been created and adopted such as DCAT-AP, GeoDCAT-AP, ADMS and Project Open Data (POD), mostly focused on Dataset description. There is no well-established profile that either generalize or specialize DCAT to describe services, schemas, schema mapping, vocabularies and portrayal information. The Linked Data Query Language SPARQL and its geospatial extension GeoSPARQL are now well-established standards to query Linked Data representation with geospatial information. The SPARQL query Protocol provides a partial answer to access DCAT-related data but it is not sufficient and sometimes too complex to manage and access catalog information through a REST API. CKAN API, an open-source software powering many Open Data Portal have provided recently a plugin to export data in DCAT format. As there is more and more data published in DCAT format in the community, there is a need to have a simple and standardized REST API to manage and access Linked Data describing assets managed by catalogs/registries.

Furthermore, there are many clients that access web services from web browser using protocol based on JSON written in Javascript. The direct use of RDF formats such as RDF/XML, N3, Turtle has



proved that they are not easily exploitable by these clients. The new W3C standard [JSON-LD](#) is designed to address this gap. It provides the ability to describe Linked Data in JSON and vice versa by leveraging the JSON-LD Context.

The use of REST API in the web community has been widely successful because it aligns well with web principles and enables rapid integration to build web-based application. A number of effort related to Linked Data and REST API have been conducted such as [Linked Data Platform 1.0 \(LDP\)](#). Recent trends in the industry see the emergence of Hypermedia formats (HAL, Hydra, Collection+JSON, Siren) to support data REST API with hypermedia controls. This introduces a level of decoupling between server and client ecosystem that could enable future evolution of APIs without breaking client ecosystem. The challenge of this testbed is to find out how REST, Hypermedia API, Linked Data representation and APIs, JSON-LD can be combined together to meet the requirements of the service.

### 6.1.2. Semantic Mediation Service

During the OGC Testbed 11, Image Matters LLC developed the first iteration of the Semantic Mediation Service, demonstrating the transformation of Homeland Security Working Group (HSWG) Incident Ontology to the Canadian Emergency Management Symbology (EMS) Ontology, using a rule engine. The engine was based on the **Semantic Mediation Ontology** and **SPARQL Extension ontology** expressing rules and functions of transformations between two semantic models (expressed as ontologies). The transformation from a source ontology to target ontology is called **Alignment** in the Semantic mediation Ontology.

While semantic transformation in the Testbed 11 was demonstrated on Linked Data representation, there is a large number of messages or documents in the current OGC standards that are based on XML representation. Their structures and syntaxes are defined with XML Schemas (GML, ISO 19139, CSGDM, NMIS). The messages often need to be converted to other XML representation based on another XML schema, typically using XSL Transformations but also by scripts. There is a need to search and discover schema definitions and existing schema mappings and finding optimal transformation paths between two schemas and perform these transformations on demand. The current OGC standards do not provide standard profile to represent schemas and schema mappings in existing OGC Catalog Services. Similarly, there is no ontology or DCAT profile capable to describe semantically Schema and Schema Mappings.

There is no existing REST service that enables the search and discovery of schemas and schema mapping, performs validation, calculate optimal transformation paths and perform transformation of message from a schema to another.

### 6.1.3. Semantic Portrayal Service

The current OGC standards related to Portrayal (Style Layer Descriptor and Symbol Encoding) are based on XML schemas technologies. Styles definition, rules, symbolizers and graphics are defined XML document and do not provide global reusable identifier that allow the reusability and linking of portrayal information to emerge a "web of portrayal information". The standard ISO 19117 that describe portrayal information define an abstract model is mostly designed to be implemented in code but is less adequate to be used a descriptive model.

During the testbed 11, an initial set of portrayal ontologies were defined to describe semantically

styles, portrayal rules, point-based symbols and graphics. The information were made accessible through a specialized REST API to be accessible by a WPS to produce SLD documents. The ontologies were limited only on point-based graphic symbols (based on PNG and True type font). The ontology was not able to model line, text and area based symbols and more advanced graphic objects and properties.

A number of attempts have been done in previous testbeds (OWS-8) to represent portrayal information in OGC catalog, but no standard profile has been defined so far and none of them has used a semantic-based approach.

There is a need to manage portrayal information in a catalog that can be related other information such as feature types, layers and taxonomies. This information can be searched and used to perform map portrayal using symbologies from different communities. The initial implementation of the Semantic Portrayal Service in the OGC Testbed 11 did not provide rendering endpoint for the symbology.

## 6.2. Requirements Statement

### 6.2.1. Semantic Registry Service

The following were the initial four general RFP requirements on the DCAT as a REST service implementation.

- It shall be evaluated how DCAT can describe the same service and data sets in RDF as the other catalog services do using XML Schema instance documents compliant to ISO 19115.
- Demonstrate what role DCAT can play as a heterogeneous catalog integration mechanism and as a possible simplification of the setup and use of catalogs.
- The DCAT REST implementation shall serve as a Semantic Portrayal Catalog. The Semantic Portrayal Catalog uses an ontology model for managing styles and provides interfaces to access,create,read,update,and delete styles.
- The DCAT as a REST service shall interface with the Schema Registry. The Schema Registry enables the discovery of XML Schemas, tranformation logic, and ontologies. These items shall be served by the DCAT as a service implementation.

From these initial high-level requirements, we derived and added the following ones:

- Definition of Common Core Model to represent datasets, services, portrayal information, schema and schema mappings.
- Demonstrate usage of Linked Data to link heteregeneous Registry Objects
- Definition of REST API
- Accomodate Web Client producing and consuming JSON.
- Ability to evolve API by favoring decoupling between server and client
- Manage multiple registers (Dataset,Schema, Portrayal Registers)
- Harvesting API from multiple catalogs (CSW ebRIM, CSW ISO and more)

## 6.2.2. Semantic Mediation Service

The following items describes the requirements for the Semantic Mediation Service:

- SRIM Profile for schema and schema mapping
- Semantic Registry as a service shall interface with the Schema Registry which enables the discovery of XML Schemas, transformation logic, and ontologies.
- Support of XML Schema and XSL Transformation
- Harvesting of Schema and Schema mapping from CSW eBRIM
- Representation of schema and schema mapping using Linked Data representation
- Definition of REST API
- Validation of Document against Schema
- Transformation from document from Schema A to Schema B.
- Transformation chaining

## 6.2.3. Semantic Portrayal Service

The following items describes the requirements for the Semantic Mediation Service

- Refinement Portrayal ontologies and introduction of Graphic ontology to support line, area, text and composite symbols.
- Ability to convert of SLD to Semantic Representation
- The Semantic Registry implementation shall serve as a Semantic Portrayal Catalog. The Semantic Portrayal Catalog uses an ontology model for managing styles and provides interfaces to access, create, read, update, and delete styles.
- Reusability and linking to existing definition (rules, styles, graphic, symbolizers)
- REST API for Rendering of Symbols
- REST API for Rendering of Data using a given style
- Ability to integrate REST API with Web-based client using JSON.
- Ability to use portrayal on multiple data representation (Linked Data, GML, JSON).

# Chapter 7. Solutions

This section summarizes the solutions that have been envisioned at the beginning of the testbed, experimented with during the testbed, and that have either been discarded, or implemented, or the decision has been deferred to future activities.

## 7.1. Targeted Solutions

This first section addresses all solutions that have been discussed during the testbed. They include those possible solutions that have been discarded during the testbed.

### 7.1.1. Semantic Registry Service Targeted Solutions

The initial requirements of the testbed was to implement a **DCAT REST Service** that enables the access to DCAT Dataset information. However it quickly emerged that the integration of ISO 19139, NMIS, Services, Schema, Schema Mapping and Portrayal information in the service was not possible using DCAT alone, as DCAT is mostly focused on Dataset description. More generalization was needed to accommodate the different types of information objects in the future (for example Map, Layer, Vocabulary, Sensor).

The closest standard that provides an extensible framework to represent different business objects is the Electronic Business eXtensible Markup Language (ebXML) Registry Information Model (ebRIM). The ebXML registry describes objects that reside in a repository for storage and safekeeping. The information model does not deal with the actual content of the repository. All elements of the information model represent metadata (data type and data relationships) about the content stored in the repository. Such information is used to facilitate ebXML-based Business-to-Business partnerships and transactions. The registry information model provides a high-level schema for the ebXML registry. The ebRIM model has a lot overlapping constructs with the ones defined in RDF and OWL. However we found out that the definition of the classes and properties in ebRIM were often not well aligned with well-established ontologies (with the exception of Dublin Core Terms) and the best practices in the Linked Data community. ebRIM introduces a lot of verbosity to describe simple graph structure thus making very hard to build queries that naturally matches the graph structure. The Linked Data model and the standard SPARQL query language provides a more modern, decentralized approach that lowers the bar of integration and learning curve of manipulating graph-oriented data structure and seems a better match to favor reusability and linkage of information.

By analyzing the ebRIM Model and the ISO 19135 standard, we decided to create a superset of DCAT that support different types of registry **items**. We borrowed the notion of **registers** and **items** from both standards. The proposed solution is a new service called **Semantic Registry Service** that manages items described semantically and can perform semantic enrichment to better enable search and discovery for information.

#### Semantic Registry Information Model

We conducted a comparative analysis between different standards related to ISO 19115 profiles and DCAT-related standards (DCAT, DCAT-AP, ADMS, GeoDCAT-AP, Project Open Data). The goal of this analysis was to evaluate how well DCAT can describe the same registry objects (services, datasets,

schema, schema mappings and portrayal information) in RDF as the other catalogue services do using XML Schema instance documents. The crosswalk was informed by the work done by the European Commission on developing a geospatial profile for DCAT (alias [GeoDCAT](#)). The metadata model provided by DCAT includes classes and attributes for identifying and describing catalogues, datasets, catalogue records, publishing agents and distribution. The metadata model does not include any classes or attributes for identifying or describing services. The absence of classes and attributes for service metadata, portrayal information, schemas and schema mappings in DCAT meant that several ISO service metadata elements did not have equivalent fields in DCAT to map to. To fill the gaps, we determined that a superset of DCAT was needed.

The testbed designed a **Semantic Registry Information Model (SRIM)** by generalizing the DCAT model to include concepts from ISO 19135, the international standard for procedures for item registration in geographic information systems and core metadata needed to express a large variety of information objects and enable better search and discovery.

### **Semantic Registry REST API**

To facilitate the integration with web-based clients, we decided to implement a REST API that primarily supports JSON\_LD output format. However we support also Linked Data Formats for item and register descriptions to enable machine to machine integration. The choice of JSON-LD was based on the fact that it provides a bridge between Linked Data and JSON.

It was also decided that the Semantic Registry REST API will implement both Level 2 and Level 3 (hypermedia REST API) on the Richardson Maturity Model so it can accommodate existing frameworks (such as AngularJS) that build REST API by constructing URL based on well defined url patterns on the client side. The Hypermedia REST API uses the Hypermedia Application Language as it has widespread adoption in the community and demonstrates well the use of hypermedia-control within JSON, and how REST API can evolve independently from the client ecosystem without breaking compatibility.

### **Semantic Registry Integration with Multi-Catalogs**

A number of approaches were considered to integrate multiple catalogues services using different protocols and models (CSW 2.0, 3.0 and eBRIM) with the semantic registry. The different approaches considered during this testbed are described in details in Testbed-12 Catalogue and SPARQL Engineering Report[OGC 16-062].

For practical reasons and due to limited timeframe, we choose to implement a harvester service that convert records from different catalogs to the SRIM model and store them in the Semantic Registry Repository. We also decided to map only the elements of information that were relevant for search and discovery. We also chose to postpone the issue of synchronization of sources for future testbeds. Publish/Subscribe protocols will need to be investigated in the context of the semantic registr in future testbeds.

### **7.1.2. Semantic Mediation Service Targeted Solutions**

The focus of Semantic Mediation Service for this testbed was to focus on transformation of XML document expressed in XML schema using transformation based on XSLT. One of the requirement was to define a eBRIM profile to represent Schema and Schema Mapping served by aeBRIM CSW

implementation and integrate it with the Semantic Registry. A review of the existing standards to represent schemas and schema mappings in a registry was conducted. Only DCAT and ADMS were found relevant but no profile were defined to accommodate the specificities of schemas and schema mappings such as source and target schema in a mapping. A review of the ebRIM model in CSW was also performed and concluded that an extension was needed to represent schema and schema mapping information in the ebRIM model.

Instead of defining a separate service that manages the schema and schema mapping separately, we decided to extend the core Semantic Registry Information Model by creating a profile to represent them semantically. The benefit of this approach is to test how well the core model can be extended and how well the Semantic Registry REST API can be reused to accommodate different domain models and convenience service APIs. We decided to implement the Semantic Mediation Service as convenience service on top of the Semantic Registry that performs validation and transformation between schemas (including finding a chain of transformation between two schemas). The Semantic Mediation Service would delegate the CRUD operations for schemas, schemas mappings to entirely to the Semantic Registry. We also decided to use a REST API that provide hypermedia controls with JSON-LD representation to lower the bar of integration with web clients, as well as getting Linked Data representation (RDF/XML, Turtle) to describe schemas and schema mappings for machine to machine support.

### 7.1.3. Semantic Portrayal Service Targeted Solutions

The focus of Semantic Portrayal Service for this testbed was on storing and accessing portrayal information managed by the semantic registry to support symbology mediation and rendering. Instead of defining a separate service that manages the portrayal information separately, we decided to extend the core Semantic Registry Information Model by creating a profile to represent this information semantically. The benefit of this approach is to test how well the core model can be extended and how well the Semantic Registry REST API can be reused to accommodate different domain models. We decided to implement the Semantic Portrayal Service as convenience service on top of the Semantic Registry that performs portrayal information search and rendering. The Semantic Portrayal Service would delegate the Creation/Update/Delete operations for portrayal information entirely to the Semantic Registry. We also decided to use a REST API that provide hypermedia controls with JSON-LD representation to lower the bar of integration with web clients, as well as getting Linked Data representation (RDF/XML, Turtle) to describe portrayal information for machine to machine support.

## 7.2. Recommendations

This second section summarizes the recommended solution(s) that will be further described in following clauses. It briefly explains the solution(s) and ideally links to relevant sections.

### 7.2.1. Semantic Registry Service Recommendations

#### Semantic Registry Information Models

To provide an extensible framework for representing information in a registry, we defined a superset of DCAT called **Semantic Information Registry Model (SRIM)** that defines the set of core classes and properties that can be used to represent any resources of a domain of interest. The core

ontology is extended by defining application profiles. The following profiles were developed during the testbed:

- **Dataset/Service Profile:** Used to describe Dataset and Services (such as the one defined in NMIS, ISO 19139). This profile is heavily based on DCAT and GeoDCAT-AP.
- **Schema Application Profile:** Used to describe Schema and Schema Mapping (which extends DCAT Dataset) and used by the Semantic Mediation Service
- **Portrayal Application Profile:** Used to describe Portrayal information such as Styles, Symbols, Portrayal Rules. This profile was used by the Semantic Portrayal Service.

### **Semantic Registry REST API**

To facilitate the integration of clients with the Semantic Registry Service, we recommended the use of REST API supporting the encoding of the SRIM profiles in Linked Data format using RDF/XML, Turtle, N-Triples and JSON-LD. We also recommended to accommodate Level 2 (Resources with HTTP Verbs) and Level 3 (Hypermedia-driven) of the [Richardson Maturity Model](#). We choose the Level 3 Hypermedia-driven API using the Hypermedia Application Language (HAL+JSON), which is gaining in popularity in the REST community.

### **Integration of Multi-Catalog REST API**

For this testbed, the Semantic Registry service harvested metadata from different OGC Web Catalogs and converted the information to SRIM profiles encoding, but we also allowed for cascading requests to other GeoSPARQL Services that implement the profiles.

## **7.2.2. Semantic Mediation Service Recommendations**

To facilitate the integration of clients with the Semantic Mediation Service, we recommended the use of REST API supporting the encoding of the SRIM Schema Application profile in Linked Data format using RDF/XML, Turtle, N-Triples and JSON-LD. We also recommended to accommodate Level 2 (Resources with HTTP Verbs) and Level 3 (Hypermedia-driven) of the [Richardson Maturity Model](#). We choose the Level 3 Hypermedia-driven API using the Hypermedia Application Language (HAL+JSON), which is gaining in popularity in the REST community. To favor reusability of functionalities, all the CRUD operations of the schemas and schema mappings were implemented by the Semantic Registry. The Semantic Mediation Service was build a convenience service on top of the Semantic Registry to provide search capabilities, validation, transformation path calculation and actual transformation of document based on the path calculated from the schema mappings managed by the registry.

## **7.2.3. Semantic Portrayal Service Recommendations**

To align better with current rendering engine implementation and current descriptive standard for Portrayal (SE, SLD), we decided to align the portrayal ontology closer to the OGC Symbol Encoding (SE)and SVG. We developed the Graphics and Symbolizer ontologies that are closely with these standards, but provide mechanism to support future extensions for more complex stlying scenarios.

The Semantic Portrayal REST Service delegates the CRUD operations on portrayal information to the

Semantic Registry which implements the SRIM Portrayal Profile. The Semantic Portrayal Service implements a REST API Level 2 and Level 3 on Richardson Maturity Model. We implemented the Level 3 Hypermedia-driven API using the Hypermedia Application Language (HAL+JSON), which is gaining in popularity in the REST community. The Semantic Portrayal Service should be a convenience service build on top of the Semantic Registry containing Portrayal information by providing search capabilities and rendering points for rendering symbol glyphs for legend and map rendering of geospatial data.



# Chapter 8. Semantic Registry Service

## 8.1. Overview

Semantic metadata plays a central role in facilitating the discovery and the assessment of geospatial assets (such as datasets, services, portrayal information, schemas, maps, layers), and the integration of these assets in a specific mission. There are a number of standards, formats and APIs that provide the metadata for these assets, but in order to perform efficient search, we need to convert this information into a unified machine readable semantic representation. It is this conversion that enables the discovery of relevant resources that satisfy the mission of the end user. As we increase our understanding of the kind of metadata information needed to perform better and smarter search, we need a model that accommodates extensions over time without breaking the proposed architecture.

During this effort, a number of metadata standards were reviewed (including W3C standard DCAT, DCAT-AP, GeoDCAT-AP, ADMS, Project Open Data 1.1, Dublin Core, ISO 19115, ISO 19119) to identify the common and relevant metadata information needed for search and discovery and to identify any additional metadata information needed to describe dataset, service, portrayal, schema and schema mapping information. It quickly emerged that the DCAT standard and its different application profiles were dataset-centric and insufficient to describe the metadata for portrayal information, schemas, and services. The goal of this effort was not to define a new standard, but to leverage the existing standards to define an application profile of DCAT, with additional properties and fields, that could accommodate the schema, schema mapping, service and portrayal information needed for enhanced search and discovery while still preserving backward compatibility with existing standards.

The effort resulted in a new ontology called **Semantic Registry Information Model (SRIM)**. SRIM is defined as a **superset of DCAT** and its existing application profiles (DCAT-AP, GeoDCAT-AP, ADMS). It introduces a superclass of **dcats:Dataset** called **srims:Item** and the notion of a **Register** (as defined in ISO 19135). The ontology draws from multiple well-established standards such as W3C DCAT, Project Open Data 1.1, DCAT-AP, GeoDCAT-AP, VCard, Dublin Core, PAV, and ISO 19115, but also addresses some gaps in the standards, such as the description of web services (for example OGC WMS, WFS), richer descriptions of geospatial data, and additional metadata to model schema, schema mapping, and portrayal information, to enable better semantic search of resources that fit with a user's mission. SRIM enables the integration of different metadata providers (CSW, CKAN, POD WAF, WMS, and WCS) by providing a **common core vocabulary** to describe resources (data, services, vocabularies, map, layers, schemas, etc.) and by accommodating the specificities of each resource by leveraging the built-in extensibility mechanism of OWL. The integration is done through the use of a semantic bridge that maps the syntactic metadata (JSON, XML based) to the semantic representation based on the SRIM model. The SRIM Core model has been extended by introducing **SRIM application profiles** to represent other kinds of geospatial assets such as schemas and portrayal information (see sections on Semantic Mediation and Semantic Portrayal Service).

The purpose of the **Semantic Registry Service** (initially referred to as **DCAT REST API**) is to define a common interchangeable metadata format for geospatial portals and a REST protocol to access this information. In order to achieve this, SRIM defines a set of classes and properties, which are

grouped into mandatory, recommended and optional. Such classes and properties aid interoperability by corresponding to information about register items and registers that is shared by many data portals. Although the Semantic Registry is designed to be independent from its actual implementation, RDF [RDF] and Linked Data [LDBOOK] are the reference technologies that perform the modeling to preserve the semantic fidelity of the conceptual model. However, we wish to facilitate a wide adoption, so we are providing an encoding based on JSON, which could be converted transparently back to a semantic model using a JSON-LD context. The JSON is closely aligned with the Project Open Data metadata schema 1.1 standard, but some extensions and modifications were made when needed to accommodate the Semantic Registry's requirements. Preferring a decoupling of the server and client ecosystem, the Semantic Registry implementation uses a **hypermedia-driven** REST API using the **Hypermedia Application Language (HAL)** with JSON-LD as the payload. Every endpoint of the REST API also provides a Linked Data representation of the resources based on the SRIM ontology.

The following sections describe the different standards that were reviewed, the SRIM model and the implementation details on both the server and the client sides. We also explain the rationale behind some of the design decisions when applicable.

## 8.2. Review of existing standards

### 8.2.1. DCAT

DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. By using DCAT to describe datasets in data catalogs, publishers increase discoverability and enable their applications to easily consume metadata from multiple catalogs. It further enables decentralized publishing of catalogs and facilitates federated dataset search across sites. Aggregated DCAT metadata can serve as a manifest file to facilitate digital preservation.

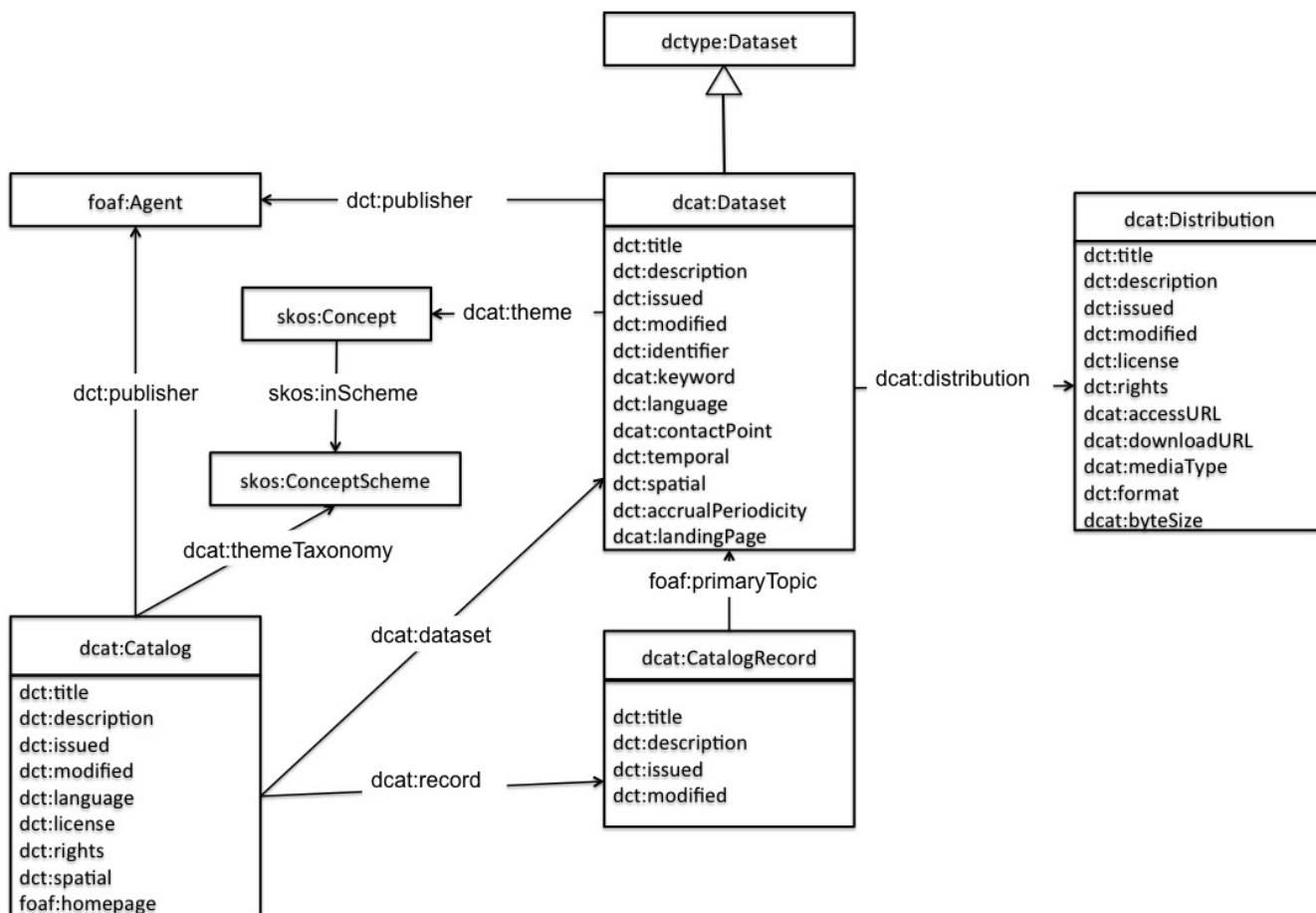


Figure 1. DCAT Model

### 8.2.2. DCAT-AP

The **DCAT Application profile for data portals in Europe (DCAT-AP)** is a specification based on the Data Catalogue vocabulary (DCAT) for describing public sector datasets in Europe. Its basic use case is to enable cross-data portal search for data sets and to allow public sector data to be easily searchable across borders and sectors. This can be achieved by the exchange of descriptions of datasets among data portals.

In February 2015, the [ISA<sup>2</sup> programme](#) of the European Commission has started an activity to revise the DCAT-AP, based on experience gained since its development in 2013. The outcome of this effort was the publication of DCAT-AP 1.1.

The [European Data Portal](#) is implementing the DCAT-AP as the common vocabulary for harmonizing descriptions of over 258,000 datasets harvested from 67 data portals from 34 countries. The DCAT-AP is used in the [Open Data Support](#) service initiated by the European Commission with the purpose of realizing the vision of European data portals.

### 8.2.3. GeoDCAT-AP

**GeoDCAT-AP** is defined as an extension of **DCAT-AP** for describing geospatial datasets, dataset series, and services. It provides an RDF syntax binding for the union of metadata elements defined in the core profile of [ISO 19115:2003](#) and those defined in the framework of the [INSPIRE Directive](#). Its basic use case is to make spatial datasets, data series, and services searchable on general data portals, thereby making geospatial information better searchable across borders and sectors. This

can be achieved by the exchange of descriptions of datasets among data portals.

### 8.2.4. Asset Description Metadata Schema (ADMS)

ADMS is a profile of DCAT that is used to describe *semantic assets* (or just 'Assets'). These assets are defined as highly reusable metadata (e.g. xml schemata, generic data models) and reference data (e.g. code lists, taxonomies, dictionaries, vocabularies) that are used for eGovernment system development.

The ADMS model is intended to facilitate federation and co-operation. Like DCAT, ADMS has the concepts of a repository (catalog), assets within the repository that are often conceptual in nature, and accessible realizations of those assets, known as distributions. An asset may have zero or multiple distributions. As an example, a W3C namespace document can be considered to be a Semantic Asset that is typically available in multiple distributions, one or more machine processable versions and one in HTML for human consumption. An asset without any distributions is effectively a concept with no tangible realization, such as a planned output of a working group that has not yet been drafted.

ADMS is an RDF vocabulary with an RDF schema available at its namespace <http://www.w3.org/ns/adms>. The original ADMS specification published by the European Commission [ADMS1] includes an XML schema that also defines all of the controlled vocabularies and cardinality constraints associated with the original document.

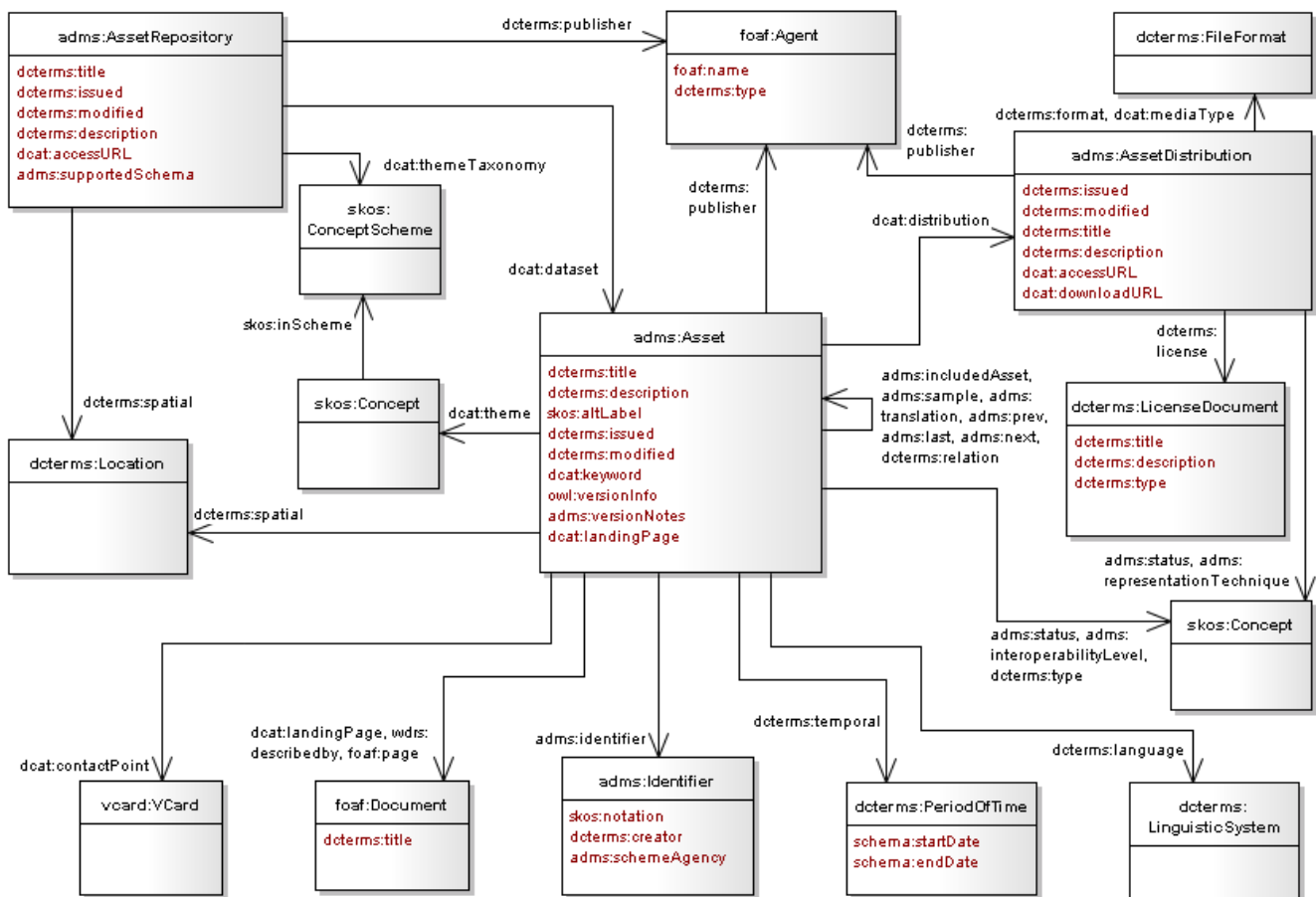


Figure 2. ADMS Model

## 8.2.5. Project Open Data (POD)

**Project Open Data** provides the implementation guide and associated resources for the Federal Executive Order on open data and data management, [M-13-13](#) “Managing Information as an Asset,” which includes the standardized metadata schema that all CFO Act agencies are required to use to publish their enterprise data inventories.

The [Project Open Data Metadata Schema](#) is a JSON-based implementation of the [W3C DCAT vocabulary](#). This standard is currently implemented by multiple data catalog platforms as well as state and local governments.

- [Project Open Data Metadata Schema v1.1](#)
- [Federal Agency Dashboard](#)
- Implemented by [city](#), [state](#), and [county](#) governments on [Data.gov/local](#)

Typically, POD documents are often published in Web Accessible Folder (WOF) and harvested by catalogs such as data.gov. The intent of POD is to lower the bar of complexity needed to represent data information by providing guidelines and recommended metadata. This enables a better search and discovery for datasets within the US government.

### 8.2.6. ISO 19115-1

The standard ISO 19115 defines the schema required for describing geographic information and services that is encoded in XML format. It provides information about the identification, the extent, the quality, the spatial and temporal aspects, the content, the spatial reference, the portrayal, distribution, and other properties of digital geographic data and services. The standard ISO 19115 is applicable to:

- the cataloguing of all types of resources, clearinghouse activities, and the full description of datasets and services;
- geographic services, geographic datasets, dataset series, and individual geographic features and feature properties.

ISO 19115-1 defines:

- mandatory and conditional metadata sections, metadata entities, and metadata elements;
- the minimum set of metadata required to serve most metadata applications (data discovery, determining data fitness for use, data access, data transfer, and use of digital data and services);
- optional metadata elements – to allow for a more extensive standard description of resources, if required;
- a method for extending metadata to fit specialized needs.

Though ISO 19115-1 is applicable to digital data and services, its principles can be extended to many other types of resources such as maps, charts, and textual documents as well as non-geographic data. Certain conditional metadata elements might not apply to these other forms of data.

ISO 19139 defines the XML-based implementation for ISO 19115. ISO 19115-1:2014 [ISO19115-1] has superseded ISO 19115:2003. At the date of publication of this document, the XML-based

implementation of ISO 19115-1:2014 (namely, ISO 19115-3), was finalised but not yet officially released.

### 8.2.7. ISO 19135

This International Standard specifies the procedures to establish, maintain, and publish registers of unique, unambiguous and permanent identifiers and meanings that are define items of geographic information. In order to accomplish this purpose, the standard specifies elements of information that are necessary to provide identification and meaning to the registered items and to manage the registration of these items.

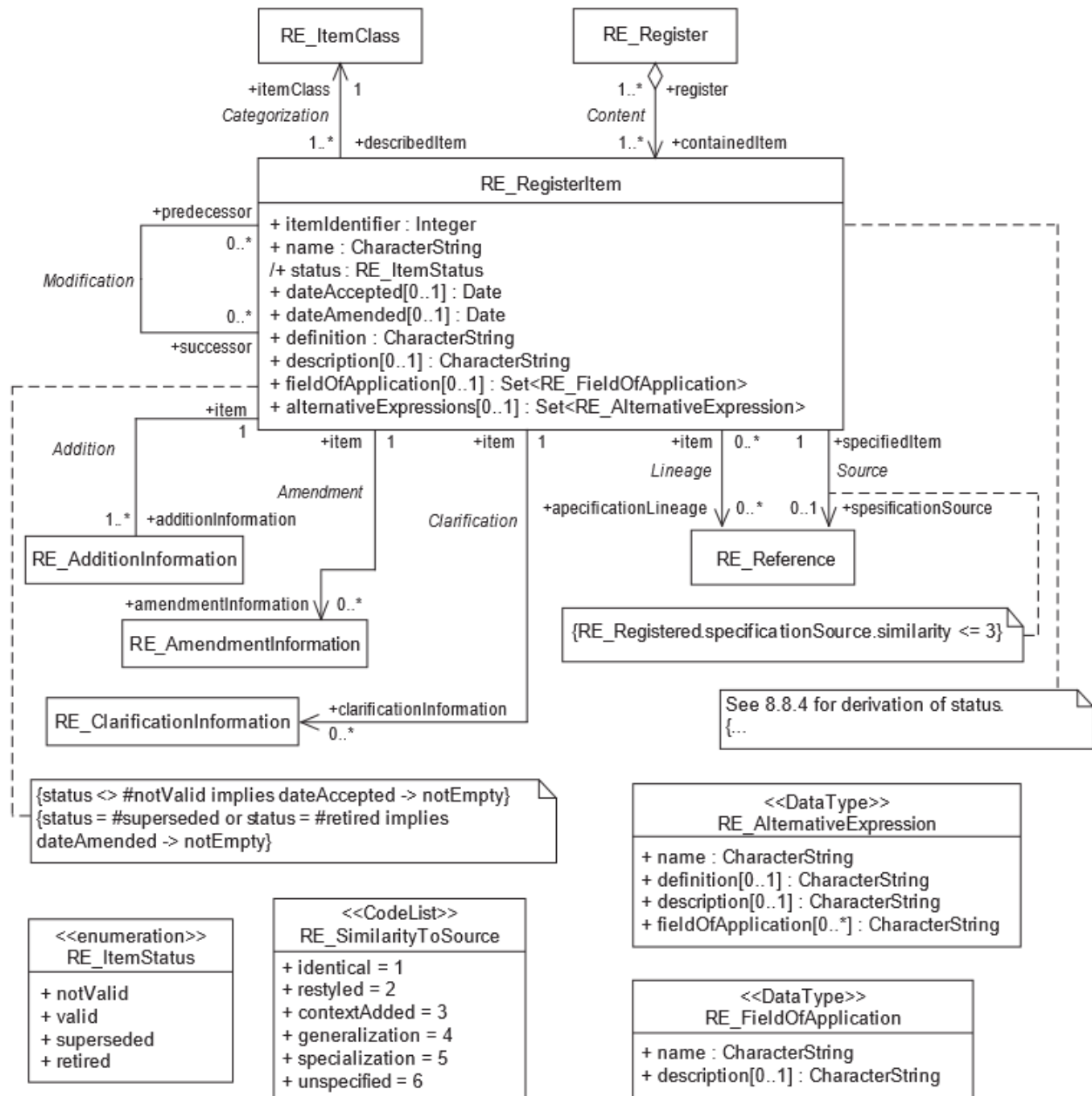


Figure 3. ISO 19135 RegistryItem

### 8.2.8. Shapes Constraint Language (SHACL)

SHACL is an RDF vocabulary for describing RDF graph structures. These graph structures are captured as "shapes", which correspond to nodes in RDF graphs. These shapes identify predicates

and their associated cardinalities, and datatypes. Additional constraints can be associated with shapes using SPARQL or other languages which complement SHACL. SHACL shapes can be used to communicate data structures associated with a process or interface, to generate or validate data, or to drive user interfaces.

Most applications that share data do so using prescribed data structures. While RDFS and OWL enable one to make logical assertions about the objects in some domain, SHACL (Shapes Constraint Language) describes data structures. Features of SHACL include:

- An RDF vocabulary to define structural declarations of the property constraints associated with those shapes.
- Complex constraints that can be expressed in extension languages like SPARQL.
- The possibility to mix SHACL shapes with other semantic web data, as SHACL is based on RDF and is compatible with Linked Data principles
- SHACL definitions represented in RDF which can be serialized in multiple RDF formats.

## 8.3. Semantic Registry Information Model (SRIM)

After analysis of the different standards, we decided to create a superset of the DCAT ontology that defines the set of classes and properties commonly used to represent any item in a register. The SRIM ontology borrows extensively from existing standards such as DCAT, GeoDCAT-AP, Dublin Core Terms, ADMS, PROV-O, PAV ontologies, and ISO 19135. In order to entice high reusability of the ontology, we decided not to enforce any restrictions in the ontology, but just define the list of properties and classes that are related through documentation (see Appendix A ). We classified the set of properties for each class as mandatory, recommended and optional.

To address different domains containing different types of items, the Core SRIM ontology is extended through **application profiles**. An Application Profile is defined as a set of classes and properties that extends the classes and properties defined in the Core Ontology. During this testbed, we defined three different profiles:

- **Dataset/Service Profile:** Used to describe Datasets and Services (such as the one defined in NMIS, ISO 19139). This profile is heavily based on DCAT and GeoDCAT-AP.
- **Schema Application Profile:** Used to describe Schemas and Schema Mappings (which extends DCAT Dataset) and used by the Semantic Mediation Service
- **Portrayal Application Profile:** Used to describe Portrayal information such as Styles, Symbols, and Portrayal Rules. This profile was used by the Semantic Portrayal Service.

We anticipate that in the future, more profiles will be defined for Maps, Layers, Coverage, Imagery, Feature Catalog, and Vocabularies.

## 8.4. Implementations

### 8.4.1. Semantic Mapping to SRIM

One of the primary functions of the Semantic Registry is to support search and discovery on a large

variety of items using a unified API. The Semantic Registry was tested to handle different item types including Datasets, Services, Schemas, Schema Mappings, as well as Portrayal information such as Symbols, and Portrayal Rules, and Feature Type Styles. To integrate the different encoding standards of this information, including ISO 19139, NMIS, ebRIM Schema Profile, and DCAT, a number of semantic mappers were implemented. These semantic mappers link each standard to the adequate SRIM profiles, and are used by harvesters to extract information from various sources of information.

We found out that a Linked Data encoding (DCAT) of information is easier to integrate than XML encoding because the latter requires code to explicitly define the mapping between the syntactic and semantic encoding. The XML encoding of information based on XML schema tends to be more unforgiving when validating data. Another advantage of using a Linked Data approach is that it favors reusability of information that can be created and managed in a decentralized way using a common encoding framework.

One of the biggest challenges when importing data into the system is the validation of the data. The RDF model provides a powerful framework to express any property of a resource by using vocabularies from different ontologies, and it can accommodate easily to partial/incomplete information. However, this flexibility causes difficulties when attempting to validate the data. Due to the limited time for this testbed, we decided to postpone the exploration of SHACL to address this issue for the next testbed. SHACL can provide a powerful way to validate data, define the shape of graph to be processed by the service.

### 8.4.2. ISO 19139 Mapping Issues.

This section summarizes the list of issues found when mapping ISO 19139 to a semantic representation with data coming from a variety of CSW sources (including data.gov and Geoplatform.gov). Some of these issues come from malformed metadata and ambiguities in the ISO 19115 standards, while others come from a lack of policies from agencies that publish metadata. These issues impede interoperability and integration of information in addition to search and discovery. The usage of Linked Data instead of XML encoding will address many of these problems, but not the ones related to policies.

#### Identification of Resources

Issue	Identification of Resources
Description	There is no consistent way of defining the identifiers for different resources (e.g. organizations, datasets, services, controlled vocabularies, etc.)
Why it is a problem?	Inability to link information and allow reusability. Resource information (concepts) are duplicated several times in different documents with variations of the same information. Updating this information is difficult to perform across all repositories. Need authoritative unambiguous references.
Recommendations	* Each resource should use a unique URI that is resolvable. * A policy needs to be put in place to manage the URI schemes of different types of resources. * The maintenance of the information for each resolvable URI should be decentralized to the authoritative party for the resource.



Issue	Identification of Resources
Benefits	A new policy to define URI Sets for US Government assets would provide a consistent means to make these trusted assets available for efficient, widespread discovery and re-use. This will encourage reuse and limit duplication.

### Resolvable URI

Issue	Resolvable URI
Description	Identifiers used in the 19139 document are often internal (e.g., a primary key in a store implementation) and not accessible as unambiguous web resources.
Why it is a problem?	The lack of consistent machine-resolvable URIs impedes interoperability and limits automation (concepts must be grounded with unambiguous meaning for services to interpret and respond). Grounded URIs will also help humans better understand important concepts.
Recommendations	* Make links resolvable and semantically-grounded URIs with the right information to support human and machine exploitation (for controlled vocabularies, licenses, organizations, etc.) * Make the information accessible for both human consumption (HTML) and machine-understanding (Linked Data).
Benefits	Enables the exploration of a “unified knowledge graph” that links and describes resources. Allows users to search, discover and navigate through “Concept Space”, whereupon each concept is resolvable to a grounded (unambiguous) resource for consistent human and machine understanding.

### Multilingual Support

Issue	Multilingual Support
Description	The current standard does not enable the support of translations of human readable text in multiple languages. Language is handled at document level, not field level.
Why it is a problem?	Users who do not understand the language of the information producer will not be able to discover relevant data for their tasks
Recommendations	Opt for an implementation that natively provides multilingual support (such as Linked data) or provide guidelines for how to handle multiple languages (e.g., through JSON protocols).

### External Resource Descriptions

Issue	External Resource Descriptions
Description	* A number of properties refer to external resources (homepage, landing page, online resource for contact, page about document, reference to metadata document). *Standards such as POD model these resources using a simple URL assigned to a property. This prevents for adding additional properties such as title, description, format or role of the document that helps the user to understand the meaning of the URL
Why it is a problem?	External resources modeled as a URL value inhibits the capture of additional information to help the role and meaning of the external (auxiliary) resource in the context of a given resource
Recommendations	* Model external resources as objects when their role is ambiguous. * If the property referring to a resource URL is unambiguous (homepage), use the URL directly.

### Invalid XLinks

Issue	Invalid XLinks
Description	For some of the ISO 19139, xlink:href are not valid URLs (example #FS Lower 48)
Why it is a problem?	The ISO 19139 documents with invalid xlink reference do not validate with a XML schema validator.
Recommendations	Comply to standard XML Schema for xlink:href using URLs
Benefits	Correct validation of ISO 19139

### Controlled Vocabulary Management

Issue	Controlled Vocabulary Management
Description	* Controlled vocabularies are not made publicly available or are not resolvable (where is the National Map Theme Thesaurus?) * Lack unique identifier for controlled vocabulary (e.g., GCMD, Global Change Master Directory) * Lack unique identifier for keyword concepts (e.g., Paris, France) * Duplication of concepts (keywords) from different taxonomies, e.g., National Map Theme Thesaurus contains “Elevation” and NGDA Portfolio Theme refers to it as “Elevation Theme”. Are they the same concept and meaning?. * Tendency to use alternative spellings for the same concept (e.g., US and United States)
Why it is a problem?	* Can’t perform semantic search * Lack consistent use of concepts (keywords) across 19139s * Ambiguity in the meaning of concepts (lack of grounded concepts)
Recommendations	* Define concepts in SKOS encoding with unique identifiers that are resolvable * Group alternate labels or translations under the same concept * Provide SKOS mappings to other vocabularies to enable semantic search across taxonomies * Make controlled vocabularies publicly available and uniquely identified with a resolvable URL.

Issue	Controlled Vocabulary Management
Benefits	* Allows reusability of controlled vocabularies * Less verbose document * Unambiguous interpretation of key concepts * Inference enabled by using standard SKOS semantics (semantic search) * Enable Multilingual search by concept

## Keywords Types

Issue	Keyword Types
Description	The list of keyword types in ISO 19115 is limited to a few categories (discipline, strata, topic, place, temporal).
Why it is a problem?	Inability to accommodate new types of concepts such as audience, function, subject, topic, etc..
Recommendations	* Provide a mechanism to extend the list of keyword types in ISO 19115 using SKOS controlled vocabularies * Define the keyword types in a controlled vocabulary to make them uniquely identifiable and resolvable * Refer to the keyword type by resolvable URL
Benefits	* Provide an extensibility mechanism to accommodate other types of concepts (Audience, Function, Purpose, etc.). * Allows reusability of keyword types

## Keyword Labeling Inconsistencies

Issue	Keyword Labeling Inconsistencies
Description	In some instances, multiple labels are encoded as one keyword (e.g., 'list of all US states' is one keyword).
Why it is a problem?	While this is fine for doing lexical-based text search, it is not sufficient when supporting semantic search, where each concept must be grounded to a unique meaning.
Recommendations	* Each keyword should refer to one concept only * In addition to a label, use a URI to refer to a concept
Benefits	* Less verbose document * Enables inference by using standard SKOS semantics

## Authority for Controlled Vocabularies

Issue	Authority for Controlled Vocabularies
Description	The ISO 19139 uses the list of topic categories in the standard ISO 19115. There is a SKOS encoding available in the European Registry located at: <a href="http://inspire.ec.europa.eu/metadata-codelist/TopicCategory">http://inspire.ec.europa.eu/metadata-codelist/TopicCategory</a> . The mapping to Semantic Registry uses this URI to reference dcat:theme.
Why it is a problem?	If no authority are responsible of the management of controlled vocabularies, the vocabularies will not be reused and risk to be duplicated.

<b>Issue</b>	<b>Authority for Controlled Vocabularies</b>
<b>Recommendations</b>	* There is a need for a registry of controlled vocabularies that are reusable across agencies. * OGC could host controlled vocabularies encoded in SKOS (currently only a GML document is hosted by the team from Inspire).
<b>Benefits</b>	The taxonomy is maintained by the authority that defines the standard and thus will favor reusability of the vocabularies among information producers.

### Place Name Consistency

<b>Issue</b>	<b>Place Name Consistency</b>
<b>Description</b>	ISO 19139 uses keywords to define place names that reference a thesaurus that is not accessible online. There is no consistent way to define place names and resolve ambiguities.
<b>Why it is a problem?</b>	The place name can be ambiguous as there are many locations with the same name (e.g. Leesburg, FL versus Leesburg, VA)
<b>Recommendations</b>	* Use unique resolvable identifier (URI) to define place name along with a human readable name. * Provide a human readable page for place name URI and Linked Data representation, with paronymy relationships, i.e., A semantic gazetteer. * Reference gazetteers with a resolvable URI. * Use well known gazetteers (Geonames, GNIS)

### Contact Point

<b>Issue</b>	<b>Contact Point</b>
<b>Description</b>	Contact Point in ISO 19139 is not systematically encoded in the document. The individual's name is required in POD but is not always present in the ISO document. A generic email reference for the contact role is sometimes used.
<b>Why it is a problem?</b>	When a problem is present in the metadata, a contact point with an email should be available for expedient resolution of issues.
<b>Recommendations</b>	* Enforce Contact Point for every Resource with email, role name, and individual name. * Email associated with contact point should be assigned to a role, not a specific individual.
<b>Benefits</b>	The use of a generic role-based email for the contact will smoothly handle staff changes.

### Responsible Party without Role

<b>Issue</b>	<b>Responsible Party without Role</b>
<b>Description</b>	Some responsible parties are published without a role, while the ISO standard indicates that the role is mandatory
<b>Why it is a problem?</b>	Without a role, we are unable to understand how each party relates to a data source.
<b>Recommendations</b>	Enforce role in ISO 10139 for each responsible party

Issue	Responsible Party without Role
Benefits	We are able to discern how each party relates to a metadata item unambiguously.

### Responsible Party Role Encoding

Issue	Responsible Party Role Encoding
Description	ISO 19139 outlines a well-defined taxonomy for Responsible Party roles (e.g., Publisher, etc). ISO 19139 refers to a GML document, through a URL and an Xpointer, which contains roles and many other concepts (instead of a unique concept)
Why it is a problem?	* Information conveyed in a GML document cannot be interpreted automatically. The XML schema needs custom code to be interpreted, and the Xpointer URL cannot be used in the context of Linked Data * In order to understand the meaning of a role, an unambiguous machine-readable description and human-readable page needs to be provided for each role.
Recommendations	Encode the role taxonomy in SKOS (machine-readable) and use resolvable URIs for roles.
Benefits	Both machine and human can understand the unambiguous meaning of the concept.

### Organization Hierarchy

Issue	Organization Hierarchy
Description	ISO 19139 does not provide support for the subOrganizationOf property (recommended by Project Open Data).
Why it is a problem?	* Difficult to understand the hierarchy between organizations * Search within a hierarchy of organizations is broken.
Recommendations	* Add a subOrganizationOf property to the ISO 19115 standard * Make the organization resolvable to a URL that provides a machine-processable definition of the organization
Benefits	When a resource search is performed for a given organization, the hierarchy can also be leveraged to search within suborganizations (using transitive inferencing).

### Inconsistent Usage of OnlineResource in ContactInfo

Issue	Inconsistent Usage of OnlineResource in ContactInfo
Description	In some documents, the link to services and distributions (zip files) is put in a responsible party's contact information (onlineResource) instead of the ServiceIdentification property or the TransferOptions in a Distribution
Why it is a problem?	The ContactInfo's onlineResource property is being misused semantically.

<b>Issue</b>	<b>Inconsistent Usage of OnlineResource in ContactInfo</b>
<b>Recommendations</b>	* Enforce a consistent way to encode distribution and service descriptions * Clarify the role of onlineResource in ContactInfo
<b>Benefits</b>	Consistency of description of services and distributions in ISO 19139, will help to make a clear distinction between service and distributed content that can be downloaded.

### Service API Standards

<b>Issue</b>	<b>Service API Standards</b>
<b>Description</b>	There isn't a consistent manner of referring to the applicable services API standard, e.g., WMS, WFS, ArcREST
<b>Why it is a problem?</b>	There is no systematic and unambiguous way to identify web services standards. The version of a standard is often not clear (OGC:WMS). Smart software, assisted by people, need to resolve spec confusion.
<b>Recommendations</b>	* Service API should reference an authoritative spec URI to remove any ambiguity. * Make the URI of the referred standard resolvable (example: <a href="http://www.opengis.net/spec/wms/1.3">http://www.opengis.net/spec/wms/1.3</a> )
<b>Benefits</b>	Proper classification of service standards, disambiguation, and support of autonomous operations

### Service API Specification

<b>Issue</b>	<b>Service API Specification</b>
<b>Description</b>	Absence of industry best practices or standards to refer to machine-processable API specifications (RAML, ALPS, Swagger, WSDL, etc.).
<b>Why it is a problem?</b>	* The ISO standard is not up to date with the techniques currently used in the industry, i.e., REST based API with machine-processable API specifications. * Specifications are defined as free text, which is not suitable for machine to machine communication.
<b>Recommendations</b>	Semantic Registry should produce a machine-processable API Document.
<b>Benefits</b>	Integration with the service API can be automated.

### Service Online Resource URL

<b>Issue</b>	<b>Service Online Resource URL</b>
<b>Description</b>	The access URL for a service is not consistently encoded. For example in a WMS, some URIs point to a GetCapabilities endpoint, while others point to the base URL of the service
<b>Why it is a problem?</b>	There is no systematic way to access the service endpoint for a given service. Software agents have to analyze the URL to get a normalized form

Issue	Service Online Resource URL
Recommendations	* Use the base URI for a service * Provide reference to a machine processable API document.
Benefits	Systematic access to a service endpoint.

### Insufficient Service Metadata

Issue	Insufficient Service Metadata
Description	The service description associated with a Dataset has minimal metadata, usually limited to an accessURL and format.
Why it is a problem	* There is not enough metadata to enable the discovery of services and the coupling of other resources to the service (layers from WMS for example) * The service identification information is sometimes too abstract to be leveraged by modern tools
Recommendations	* Use the base URI for a service * Define a rich metadata model for services and coupled resources * Provide reference to a machine processable API document or standard
Benefits	Enable the discovery of services and invocation of services in an automated way.

### Format and OnlineResource Parity

Issue	Format and OnlineResource Parity
Description	The ISO standard decouples Format and OnlineResource. One format can have more than one online resource URL.
Why it is a problem	Having multiple URLs for a format is ambiguous and not friendly to machines or users.
Recommendations	Enforce parity of OnlineResource with format.
Benefits	Proper pairing of format with online resource removes ambiguity to both machines and users.

### Download Format Versus Service

Issue	Download Format Versus Service
Description	The ISO standard does not clearly distinguish between a download file format and a service API in a Dataset distribution.
Why it is a problem	Classification of services versus downloads is difficult and not friendly to machines or users.
Recommendations	* Improve the ISO standard to make a clear distinction between a service and a download format. * Provide a rich description of services.

Issue	Download Format Versus Service
Benefits	* Enhanced classification of various distributions of datasets. * Support for autonomous operations

### Format Description

Issue	Format Description
Description	There is no consistent way to define the format of services (OGC:WMS). Usage of mime type is not consistent in the standard, and most format descriptions are not machine readable.
Why it is a problem?	Inconsistency of format description makes it difficult for software agents to access data in automatically.
Recommendations	* Use a standard URI when referring to standard service APIs * Use a MIME type from IANA to refer to representation formats.
Benefits	Enables automation, content negotiation and service selections based on controlled vocabularies.

### Insufficient Map Layer Description

Issue	Insufficient Map Layer Description
Description	The ISO standard does not provide enough information to map a dataset to a layer in a map service (WMS, ArcREST). Often multiple layers are provided by the map service and there is no deterministic way to find out which one corresponds to the dataset.
Why it is a problem?	Traceability from dataset to map layer is unavailable. The missing layer metadata is needed to support GeoPlatform search, discovery and proper use.
Recommendations	* Define a richer description of services/layers and provide them through the Semantic Registry * Define a new standard to describe layer metadata, with commensurate industry supported techniques and policies.
Benefits	Support a vastly improved layer search and map building experience.

### Data-centric Approach

Issue	Data-centric Approach
Description	Data Schema Standardization of domain models uses a syntactic approach. Imposing this strict adherence to a standard tends to minimize heterogeneity.



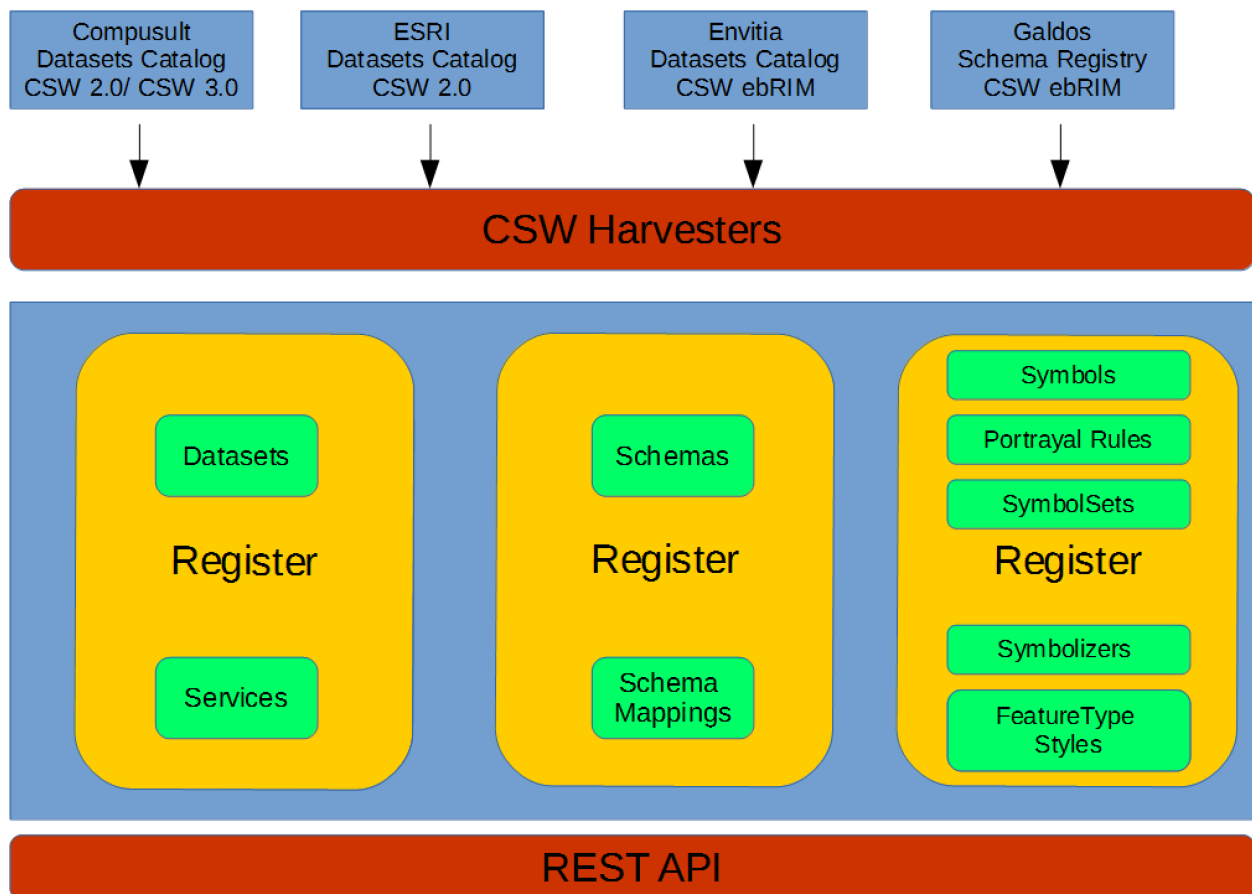
Issue	Data-centric Approach
Why it is a problem?	<p>* Data Schemas have limited expressiveness. * Data Schemas only capture the syntactic and structural constraints of a data model. This does not provide a machine-processable conceptual model or business rules. Implementations need to hardcode the rules, which risks enforcing different interpretations. * Evolution of domain model and associated software is difficult when using a data-centric approach because the business rules and data model semantics need to be hardcoded in the application. Any changes in the standard require expensive software updates. Frequent modifications of the data model require building consensus and standardization, which can be a lengthy process. * Integration and interoperability with other domains is difficult due to discrepancies between data schemas and business models, as well as the lack of common protocols, and machine-processable conceptual models and business rules.</p>
Recommendations	<p>* Use a semantic-based approach to embrace the heterogeneity of domain models by providing a common, formal, and sharable framework mechanism for easily extending metamodels to accommodate specific needs. The extensions can be done in a decentralized way without breaking the existing infrastructure. * Use of Linked Data standards (such as OWL, SHACL and SPARQL Rules) to provide a standard-based mechanism to capture formal conceptual models, along with their business rules, in a machine-processable way. Information captured in this manner, could be imported by a system implementation without writing additional code. * Use ontologies to provide a framework for extending metamodels in a decentralized way to accommodate the specificity of each domain player. The extensions can be integrated and handled by any generic-purpose semantic-based reasoner and validator without rewriting code.</p>
Benefits	<p>* Decentralized extension of the model. * Accommodation of model specificities * Shareable model and business rules that are machine processable. * Reduction of software development cost * Exchangeable machine processable rules and conceptual models, which allow automation and reduction of code. * Unambiguous interpretation of domain models * Cost reduction in software updates * Software that adapts and evolves to match changes in domain models without rewriting code. * A decentralized and organic evolution of the domain model * Software that can adapt quickly to changes in the model or business rules.</p>

### 8.4.3. Semantic Registry Service

The Semantic Registry Service was designed to manage multiple registers that are capable of containing item classes from different application profiles. To support the testbed 12, we implemented three different registers:

- **Datasets and Services Register:** Manages datasets and services collected from Compusult, Envitia and ESRI CSW instances
- **Schema and Schema Mapping Register:** Manages schemas and schema mappings harvested from Galdos CSW Schema Registry
- **Portrayal Service Register:** Manages portrayal information (styles, symbols, symbolSets, and

portrayal rules)



**NOTE**

The partitioning of the registers was done to provide some clarity in the organization of the information. However it is possible to create a register that contains multiple application profiles. The partitioning decision is based on the business requirement of the user.

These registers were populated by a **harvester service** which is integrated with the Semantic Registry Service and accessible by a hypermedia-driven REST API. The harvester service was designed to be extensible and to support multiple types of data sources, including documents extracted from a resolvable URL (Project Open Data, DCAT , ISO 19139, FDGC CSGDM documents), and advanced web services such as CSW, CKAN, Web Accessible Folder, and ESRI Web services. These plugins called **harvester types** describe the list of parameter descriptors needed by the harvester. An instance of a harvester type is called a **harvester source** and provides binding of the parameters to values. A harvester source can be triggered for harvesting manually or a given schedule, and the harvester results are returned with statistics (number of harvested objects successfully imported, number of failures) as well as the list of item identifiers. Due to limited time for implementation, only synchronous calls to harvesters are supported. Future development will handle asynchronous harvesting with on demand status reports.

The items managed by the service are stored in a NoSQL store, and are indexed and managed in a RDF store to support graph analytics and SPARQL queries.

#### 8.4.4. Semantic Registry Service REST API

The initial objective of the testbed was to provide a **DCAT REST API**, which focused on the search and discovery of **dcat:Datasets**. However, promoting the DCAT model to the superset SRIM model also necessitated a promotion of the REST API to manage registers and harvester types and sources, and to handle more general items, including Portrayal items, Schema and Schema Mapping items.

A review of existing implementations that use DCAT datasets showed that the only consensus in how to access the information through a REST API, was the use of a SPARQL query protocol. Using an OGC filter was not considered adequate enough for complex queries of RDF data, as SPARQL provided a more compact and standardized way to query linked data. One of the main considerations when designing the REST API for the Semantic Registry was to make it accessible for web clients, which primarily operate in JSON, and to bridge the gap between linked data and JSON, the Semantic Registry uses the W3C JSON-LD. The use of JSON-LD context allows the conversion of RDF models to JSON representations and vice versa. Another objective of the API was to provide a degree of separation between the server and client implementation, to allow the API to evolve in the future without breaking client ecosystems. To achieve this, the Semantic Registry uses Hypermedia Links which provide a powerful mechanism to decouple clients and servers. This corresponds to the Level 3 REST API on the [Richardson Maturity Model](#).

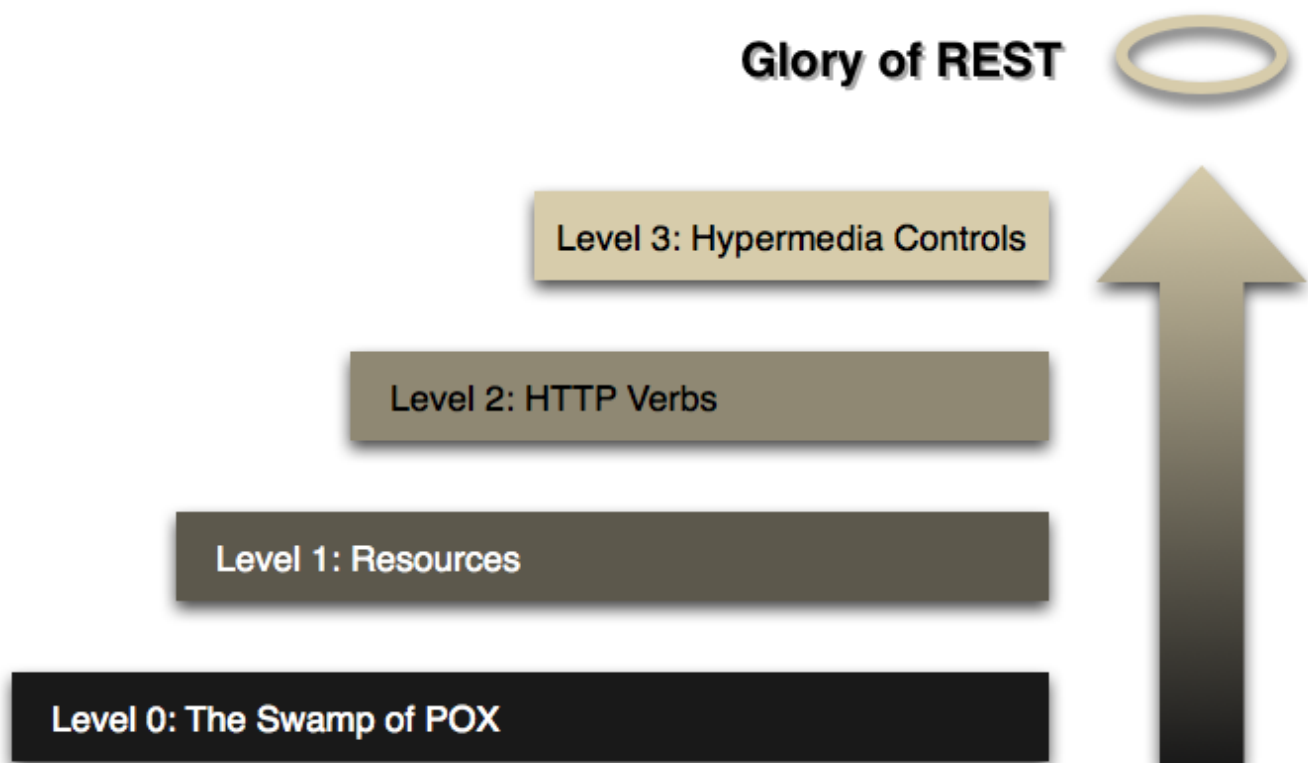


Figure 4. Richardson Maturity Model

To implement a Level 3 REST API, we adopted the IETF standard candidate [Hypermedia Application Language \(HAL\)](#), a popular standard candidate which is widely used by JSON hypermedia REST APIs.

We also acknowledge that many web frameworks (such as AngularJS) are designed for Level 2 APIs and construct URLs on the client side to access the different states of a web application. To accommodate these frameworks, we decided to also implement a Level 2 REST API by providing well-defined URL patterns to access the artifacts of the service (registers, items, harvesters types,

harvester sources) and a unique identifier for each artifact. The responses of Level 2 are identical to those of Level 3, except for the exclusion of the hypermedia links to other states. The REST API endpoints URL pattern documented in Appendix D are considered **informative only** not normative.

In addition to the Level 2 and Level 3 REST APIs that will mostly be used by web clients, we added support for Linked Data API that will mainly be used by machines. Each REST endpoint of the Semantic Registry Service also supports a Linked Data output in RDF/XML, Turtle and N-Triples formats.

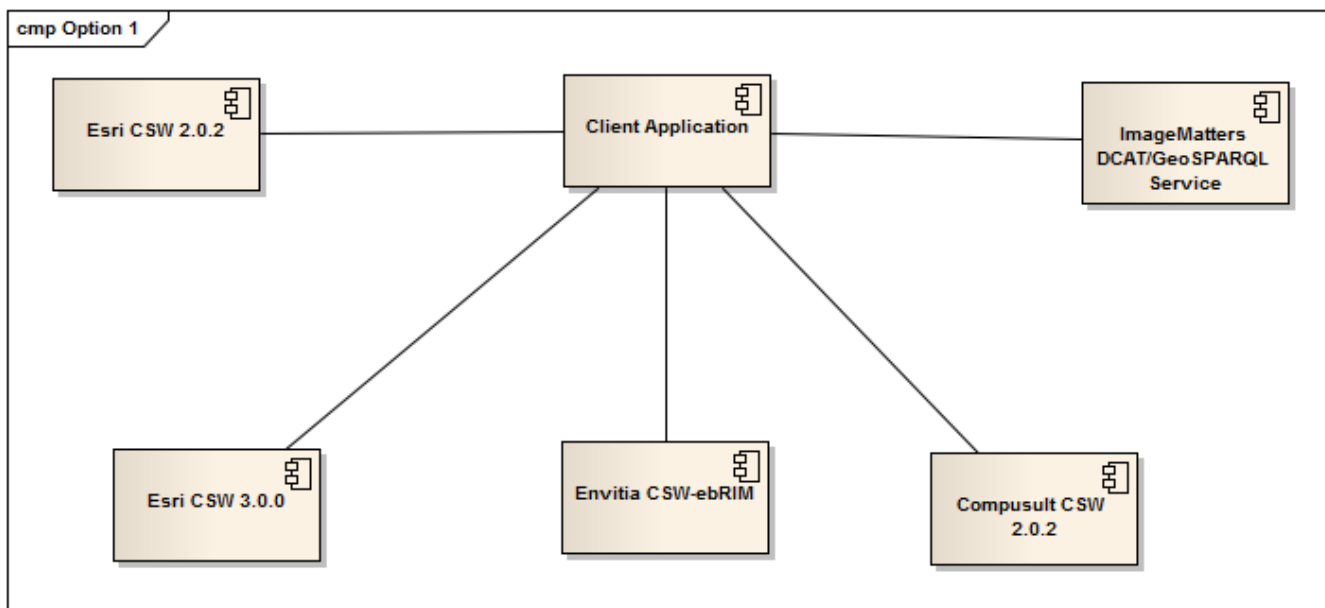
Furthermore, each Register endpoint also provides a GeoSPARQL endpoint that permits advanced SPARQL queries on the Linked Data representation of the items managed by each register.

### 8.4.5. Integration with OGC Catalog Services

To evaluate interoperability aspects in multi-catalogue type environments, the testbed considered a number of solutions. Each solution involved various types of catalogue services, for example, CSW featuring ISO based metadata and OpenSearch, other CSW offering a SOAP binding, and support for DCAT using RDF.

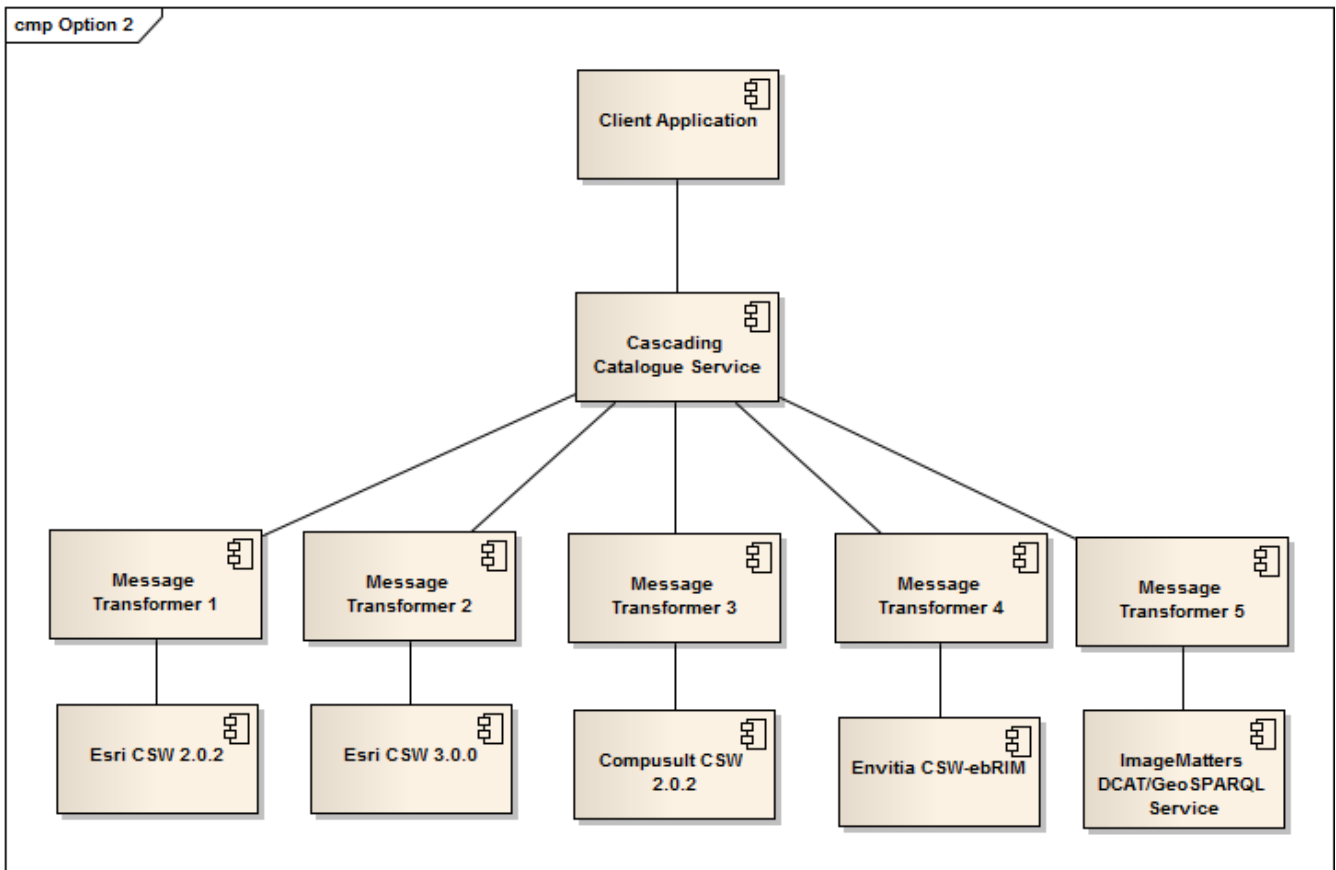
Several architectural solutions could be used to establish a multi-catalogue environment, and four key architectural solutions were identified by the Testbed. The identified solutions differ in a variety of ways, including the entry point for client applications and the computational balance between the client application and the services.

The first solution for a multi-catalogue environment includes a client application that can query the various catalogue services directly. This requires the client application to prepare appropriate queries for each catalogue service and to collate the search results when they are returned by the services.

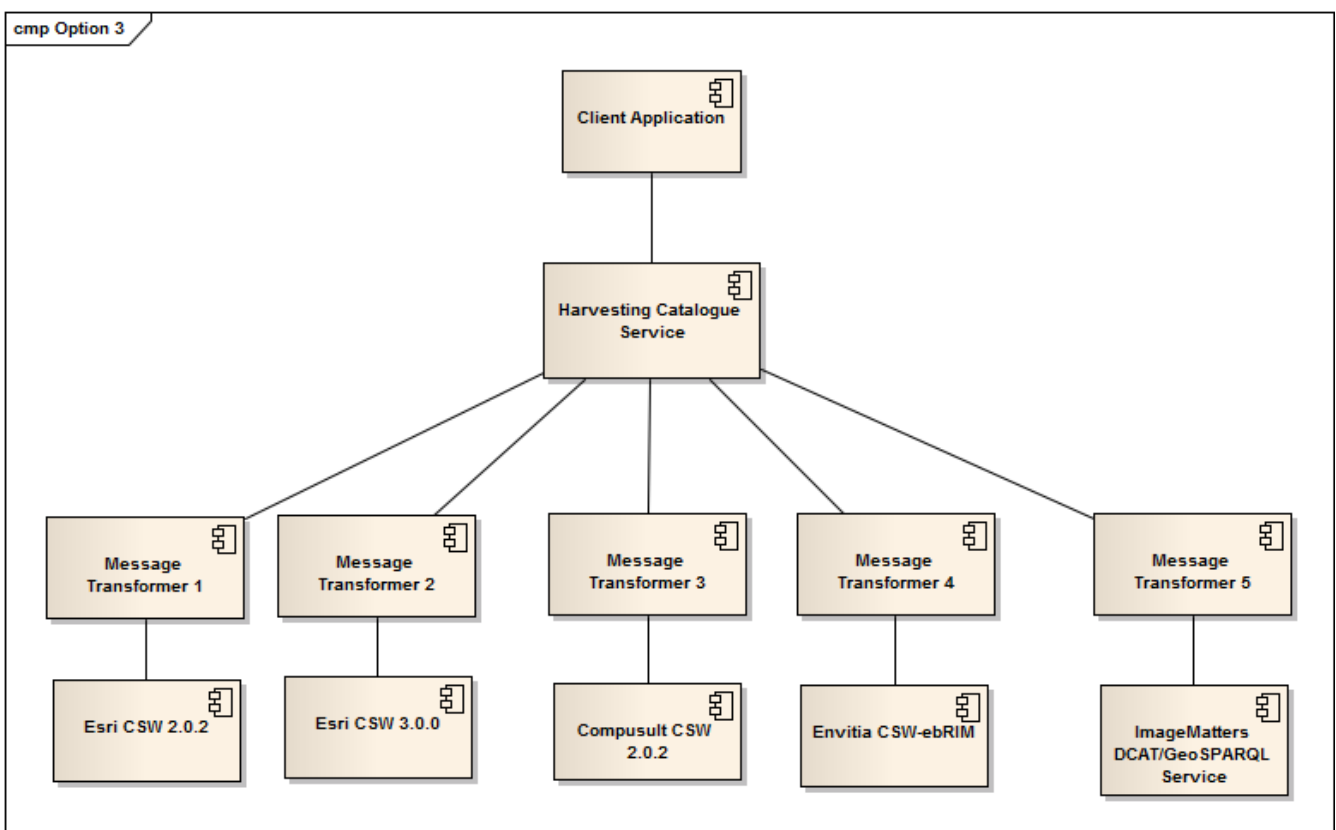


The second solution involves the selection of one of the catalogue services to initiate a distributed search. In this case, the client application only needs to prepare queries to send to the cascading catalogue service. Upon receiving a request from the client, the cascading catalogue service then adapts the request to forward to other catalogue services and returns responses from the other

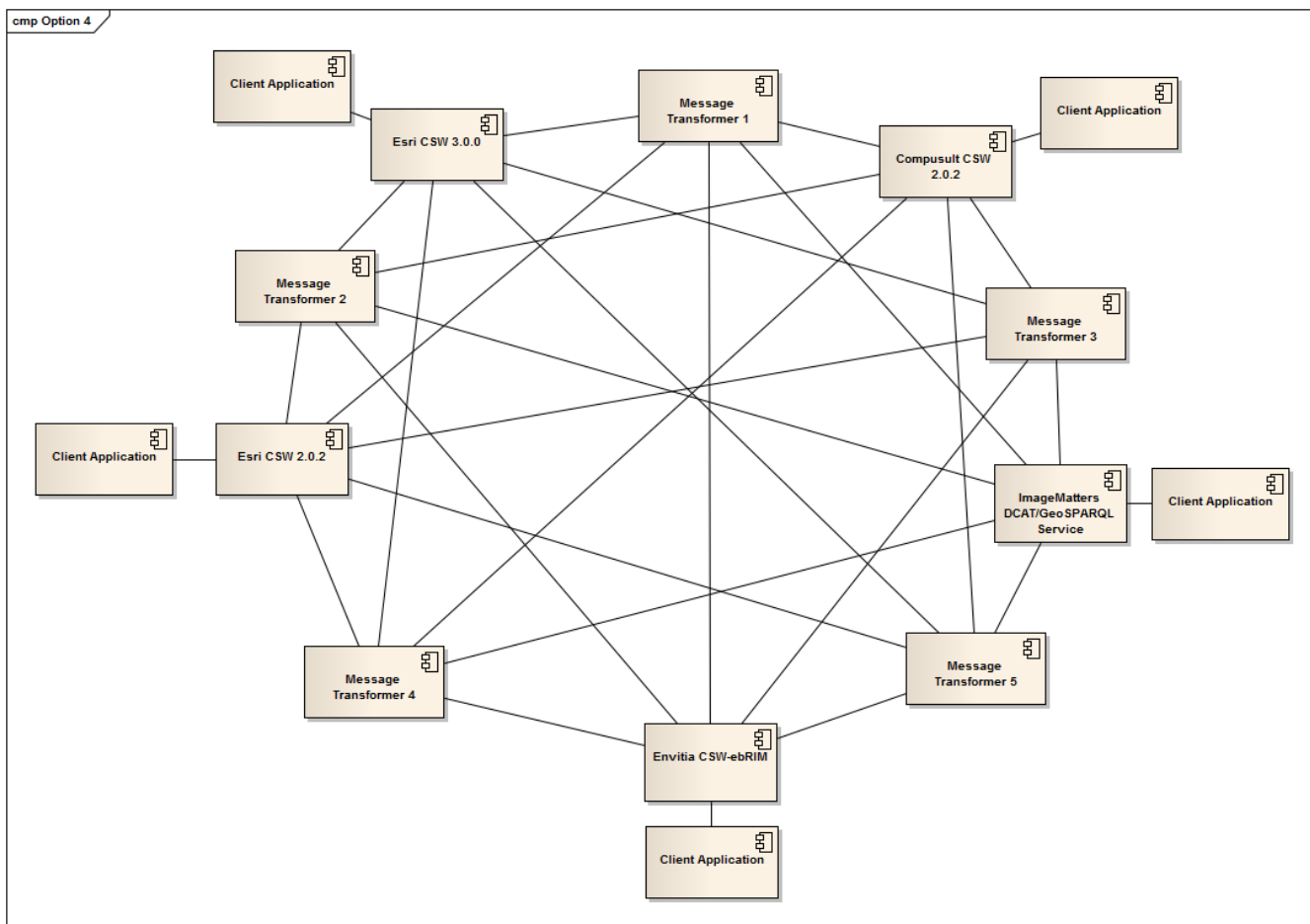
services, as well as results from its own catalogue.



The third solution involves the harvesting of metadata from one or more source catalogue services into a single target catalogue service. Harvesting is ideally conducted at a scheduled time and not when a query is received from the client. The client application can then query the target catalogue service to discover resources published by both the source and target catalogue services.



The fourth solution involves the replication of metadata between a federation of catalogue services. Replication would ideally be conducted at a scheduled time and not when a query is received from the client. The client application can then query any catalogue service to discover resources published by any catalogue service.



For this testbed, the integration with the OGC Catalog Services was accomplished by using a **Harvester Service**.

### CompuSult CSW Integration

We integrated CompuSult CSW, which serves a ISO 19139 document, using the CSW 2.0 protocol. The integration could have been done with CSW 3.0, but no open-source clients that supported the CSW 3.0 protocol were available at the time of the testbed 12. However, the Harvester configuration for CSW 3.0 would be very similar to the CSW 2.0 GetRecords operation. To map ISO 19139 to SRIM, we use a semantic mapping using the DCAT profiles. We have not found any issues validating the ISO 19139 document against their XML schema, however we found some issues in the ISO 19139 mapping (explained in Section 1.4 Implementations).

A Harvester Source for the CSW Catalog was defined and harvested on demand. The following figure shows a client displaying the harvester source for CompuSult CSW:

Semantic Registry Manager Items Search... Q

Testbed12 Compusult CSW ?

**Metadata** Modify

<b>Title</b>	Testbed12 Compusult CSW
<b>Description</b>	Compusult CSW used for OGC Testbed 12 to harvest ISO19139 documents
<b>Register</b>	<a href="#">Testbed12 Datasets Register</a>
<b>Source</b>	<a href="http://ogc-testbed12.compusult.net/wes/serviceManagerCSW/csw">http://ogc-testbed12.compusult.net/wes/serviceManagerCSW/csw</a>
<b>Harvester Type</b>	OGC CSW Harvester
<b>GetRecords XML Request</b>	<i>No GetRecords XML Request. Add one by clicking the modify button at the top of the page.</i>
<b>Resource Type</b>	<a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a>
<b>Created Date</b>	Nov 15, 2016 1:11:24 PM
<b>Last Modified Date</b>	Nov 15, 2016 1:11:24 PM

Semantic Registry Manager v0.0.0-26-gc3f1110-dirty | Image Matters, LLC

The following snippet shows the JSON encoding of the harvester source configuration:

```

{
  "id": "compusultCSW",
  "type": "csw",
  "title": "Testbed12 Compusult CSW",
  "description": "Compusult CSW used for OGC Testbed 12 to harvest ISO19139 documents",
  "created": "2016-10-03T22:49:27.311Z",
  "modified": "2016-10-03T22:49:27.311Z",
  "source": "http://ogc-testbed12.compusult.net/wes/serviceManagerCSW/csw",
  "config": {
    "resourceType": "http://www.isotc211.org/2005/gmd"
  },
  "harvestInterval": "MANUAL",
  "registerId": "datasets"
}

```

## ESRI CSW Integration

We integrated the ESRI OGC CSW, which serves ISO 19139 documents, by defining a Harvester Source with CSW 2.0. We found out that some of the ISO 19139 documents registered in the CSW were not compliant with the standards (for example missing ScopeCode in HierarchyLevel) The following snippet shows the configuration of the harvester:

```
{
  "id": "esriCSW",
  "type": "csw",
  "title": "Testbed12 ESRI CSW",
  "description": "ESRI CSW used for OGC Testbed 12",
  "created": "2016-11-15T18:11:24.203Z",
  "modified": "2016-11-15T18:11:24.203Z",
  "source": "http://gptogc.esri.com/geoportal/csw",
  "config": {
    "resourceType": "http://www.isotc211.org/2005/gmd"
  },
  "harvestInterval": "MANUAL",
  "registerId": "datasets"
}
```

### Envitia CSW Integration

Envitia provided a CSW instance with a eBRIM profile. We configured the harvester to collect dataset metadata stored in an object of type: *urn:ogc:def:ebRIM-ObjectType:OGC-I15::DataMetadata*. The following snippet shows the configuration of the CSW Harvester:



```

{
  "id": "envitiaCSW",
  "type": "csw",
  "title": "Testbed12 Envitia ebRIM CSW",
  "description": "Envitia CSW used harvest ebRIM datasets records",
  "created": "2016-11-15T18:11:24.236Z",
  "modified": "2016-11-15T18:11:24.236Z",
  "source": "http://86.188.147.99:9080/RegistryService/registry",
  "config": {
    "requestXML": "<?xml version='1.0' encoding='UTF-8'?>
      <csw:GetRecords xmlns:env-
ebrim='http://www.envitia.com/schemas/georegistry/ebrim-ext'
xmlns:xmime='http://www.w3.org/2005/05/xmlmime'
xmlns:dct='http://purl.org/dc/terms/'
xmlns:csw='http://www.opengis.net/cat/csw/2.0.2'
xmlns:gml='http://www.opengis.net/gml'
xmlns:wrs='http://www.opengis.net/cat/wrs/1.0'
xmlns:ows='http://www.opengis.net/ows'
xmlns:ogc='http://www.opengis.net/ogc'
xmlns:dc='http://purl.org/dc/elements/1.1/'
xmlns:xlink='http://www.w3.org/1999/xlink'
service='CSW' version='2.0.2'
resultType='results'
outputSchema='urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0'
startPosition='1' maxRecords='50000'>
  <csw:Query typeName='wrs:ExtrinsicObject_coi'>
    <csw:ElementSetName
typeName='coi'>full</csw:ElementSetName>
    <csw:Constraint version='1.1.0'>
      <ogc:Filter>
        <ogc:PropertyIsEqualTo>
          <ogc:PropertyName>$coi/@objectType</ogc:PropertyName>
          <ogc:Literal>urn:ogc:def:ebRIM-
ObjectType:OGC-I15::DataMetadata</ogc:Literal>
        </ogc:PropertyIsEqualTo>
      </ogc:Filter>
    </csw:Constraint>
  </csw:Query>
</csw:GetRecords>",
    "resourceType": "urn:ogc:def:ebRIM-ObjectType:OGC-I15::DataMetadata"
  },
  "harvestInterval": "MANUAL",
  "registerId": "datasets"
}

```

## CSW ebXML Schema Registry

During the testbed, Galdos provided a CSW 2.0 instance which implemented the ebRIM profile. We extended the profile to accommodate representations of Schemas and Schema Mappings. In addition, we implemented a Semantic Mapper that converts the Schema and Schema Profile to the SRIM Schema Application Profile, and integrated it with a Semantic Registry harvester.

The following shows the Harvester Source Configuration needed to access the Schemas and Schema Mappings from the CSW:

```
{
  "id": "galdosCSW1",
  "type": "csw",
  "title": "Schema Harvester from Galdos ebRIM CSW ",
  "description": "This source harvests schemas stored in ebRIM Model",
  "created": "2016-10-03T22:49:27.542Z",
  "modified": "2016-10-03T22:49:27.542Z",
  "source": "http://ows.galdosinc.com/indicio/query",
  "config": {
    "requestXML": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
      <csw:GetRecords xmlns:env-
ebrim=\"http://www.envitia.com/schemas/georegistry/ebrim-ext\"
xmlns:xmime=\"http://www.w3.org/2005/05/xmlmime\"
      xmlns:dct=\"http://purl.org/dc/terms/\"
xmlns:csw=\"http://www.opengis.net/cat/csw/2.0.2\"
      xmlns:gml=\"http://www.opengis.net/gml\"
xmlns:wrs=\"http://www.opengis.net/cat/wrs/1.0\"
      xmlns:ows=\"http://www.opengis.net/ows\"
      xmlns:ogc=\"http://www.opengis.net/ogc\"
xmlns:dc=\"http://purl.org/dc/elements/1.1/\"
      xmlns:xlink=\"http://www.w3.org/1999/xlink\"
      service=\"CSW\"
version=\"2.0.2\" \r\n\tresultType=\"results\" outputSchema=\"urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0\"
      startPosition=\"1\" maxRecords=\"50\">
      <csw:Query
typeNames=\"wrs:ExtrinsicObject\">
```

```

<csw:ElementSetName>full</csw:ElementSetName>
    <csw:Constraint version="1.1.0">
        <ogc:Filter>
            <ogc:PropertyIsEqualTo>
                <ogc:PropertyName>@objectType</ogc:PropertyName>
                <ogc:Literal>urn:ogc:def:ebRIM-ObjectType:OGC:Schema</ogc:Literal>
            </ogc:PropertyIsEqualTo>
        </ogc:Filter>
    </csw:Constraint>
</csw:Query>
</csw:GetRecords>",
    "resourceType": "urn:ogc:def:ebRIM-ObjectType:OGC:Schema"
},
"harvestInterval": "MANUAL",
"registerId": "schemas"
}

```

#### 8.4.6. Integration with Clients

A number of clients were successfully integrated with the Semantic Registry, as illustrated by the following figure:

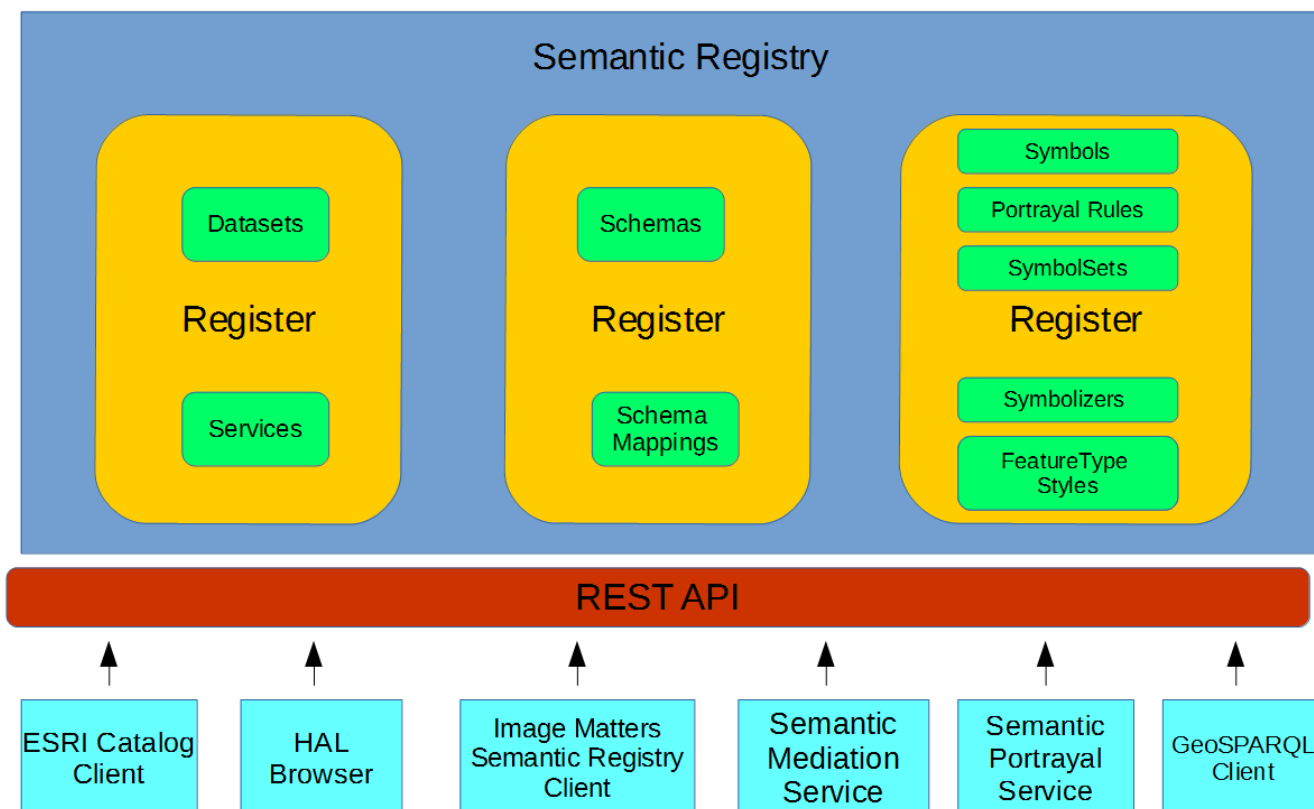


Figure 5. Overview of the Semantic Registry Clients

## ESRI Semantic Registry Client

The ESRI Client provides a plugin framework to access a variety of catalog services. For this testbed, ESRI developed a plugin to access the Semantic Registry. The following figure shows the results of a search in the ESRI Semantic Registry Client:

The screenshot displays the ESRI Semantic Registry Client interface. At the top, there is a navigation bar with links for 'Cart', 'Login', 'Register', 'Help', 'About', and 'Feedback'. Below this is a red navigation bar with 'HOME', 'SEARCH', 'BROWSE', and 'LAUNCH MAP VIEWER'. The main content area is titled 'Search' and features a search input field containing 'WMS' and a 'Search' button. Below the search bar, there are options for 'Records shown from' and 'Additional Options'. The 'WHEN' section includes radio buttons for 'Intersecting' (selected) and 'Fully within', along with 'Start Date' and 'End Date' input fields. The 'WHERE' section includes radio buttons for 'Anywhere' (selected), 'Intersecting', and 'Fully within'. A 'Text:' input field is also present. On the right side, the search results are displayed as a list of 10 items, including 'Map Unit Polygons - Large Scale', 'gem\_point: Thermokarst depression, small - airphoto observation', 'gem\_line: Limit of submergence, glaciomarine - approximate', 'Earth Material #2', 'Iceberg Open Water Polygons for the week of 20150622', 'Iceberg Polygons for the week of 20141208', 'Iceberg Polygons for the week of 20141020', 'Iceberg Open Water Polygons for the week of 20150126', 'Iceberg Open Water Polygons for the week of 20141208', and 'gem\_line: Bedrock scarp'. At the bottom of the results, there are links for 'See results through REST' and 'API: GEORSS ATOM HTML FRAGMENT KML JSON DCAT CSV'. A world map is shown on the left side of the results area, with 'Sources: Esri, DeLorme,...' and the 'esri' logo at the bottom. Below the map are links for 'Zoom To Results' and 'Zoom To Search Area'.

Figure 6. ESRI Semantic Registry Client

## Image Matters GeoSPARQL Semantic Registry Client

To perform a GeoSPARQL query against registers, the Semantic Registry was deployed with a GeoSPARQL client that performs SPARQL validation and execution, and displays the results in a table, pivot table, and a Google Chart if the data table structure is applicable. The following example performs a simple SPARQL query to list schema mappings with title, description, source and target schema:

## Semantic Registry SPARQL Endpoint

```
1 PREFIX dct: <http://www.d3.org/ns/dct#>
2 PREFIX dct: <http://purl.org/dc/terms/>
3 PREFIX schema: <http://www.opengis.net/ont/testbed12/srin/profile/schema#>
4
5 SELECT DISTINCT ?schema ?title ?description ?srcSchema ?targetSchema
6 WHERE {
7   ?schema a schema:SchemaMapping;
8           dct:title ?title;
9           dct:description ?description;
10          schema:sourceschema ?srcSchema;
11          schema:targetschema ?targetSchema
12 }
13
14 LIMIT 4
15
```

schema	title	description	srcSchema	targetSchema
<a href="http://www.opengis.net/testbed12/schemas#MINS2DDMS">http://www.opengis.net/testbed12/schemas#MINS2DDMS</a>	NMIS 2.2 to DDMS SchemaMapping	Schema Mapping from NMIS 2.2 to DDMS	<a href="http://www.opengis.net/testbed12/schemas#NGA_STND_0018_2_2">http://www.opengis.net/testbed12/schemas#NGA_STND_0018_2_2</a>	<a href="http://metadata.dod.mil/mdr/ns/DDMS/2.0">http://metadata.dod.mil/mdr/ns/DDMS/2.0</a>
<a href="http://www.opengis.net/testbed12/schemas#FGDC_TO_ISO_19139Mapping">http://www.opengis.net/testbed12/schemas#FGDC_TO_ISO_19139Mapping</a>	FGDC to ISO 19139 SchemaMapping	Schema Mapping from FGDC_STD_001_1998 to ISO 19139	<a href="https://www.fgdc.gov/schemas/metadata/fgdc-std-001-1998">https://www.fgdc.gov/schemas/metadata/fgdc-std-001-1998</a>	<a href="http://www.isotc211.org/2005/gmd">http://www.isotc211.org/2005/gmd</a>

Figure 7. GeoSPARQL Semantic Registry Client

## HAL Browser

The HAL browser was deployed alongside the Semantic Registry to demonstrate the Hypermedia REST API implementation. The HAL browser landing page uses the entry point of the services and provides UI interactions to perform valid operations (GET, POST, PUT, DELETE) on the endpoint exposed by the hyperlinks. The UI shows the embedded items in the collection and the instance in JSON. The following figure shows the entry point of the service:

The HAL Browser interface is divided into several sections:

- Explorer:** Shows the current resource path as `/registry` with a 'Go!' button.
- Custom Request Headers:** An empty text area for entering headers.
- Properties:** Displays the JSON representation of the resource:

```
{
  "type": "iri:Service",
  "title": "Semantic Registry Service",
  "description": "Semantic Registry Service prototype for OSC Testbed12",
  "category": [
    "http://www.opengis.net/specs/testbed12/semanticRegistry"
  ],
  "publisher": [
    {
      "name": "Image Matters LLC",
      "uri": "http://www.imagemattersllc.com",
      "type": "org:Organization"
    }
  ],
  "version": "0.1"
}
```
- Links:** A table of hypermedia links for various operations:

rel	title	name / index	docs	GET	NON-GET
registry:capabilities			■	▶	■
registry:items			■	▶	■
registry:sparql			■	▶	■
registry:jsonldContext			■	▶	■
registry:registers			■	▶	■
registry:harvester			■	▶	■
self			■	▶	■
curies	name: registry		■	▶	■
- Inspector:** Shows the response headers:

```
200 OK
Date: Sat, 29 Oct 2016 17:01:35 GMT
Server: nginx/1.11.4
ETag: "0918af3ef0c090ac099f0bc03f0139aa4"
Content-Length: 1158
Content-Type: application/hal-json;charset=UTF-8
```
- Response Body:** Displays the JSON response body, which is identical to the one shown in the Properties section.

Figure 8. Semantic Registry HAL Browser

## Image Matters Semantic Registry Client

To support the demonstration of the Semantic Registry, Image Matters developed a client to manage registers and harvester sources, and to perform faceted search and discovery of items. The following figure shows the search page for items with different facets:

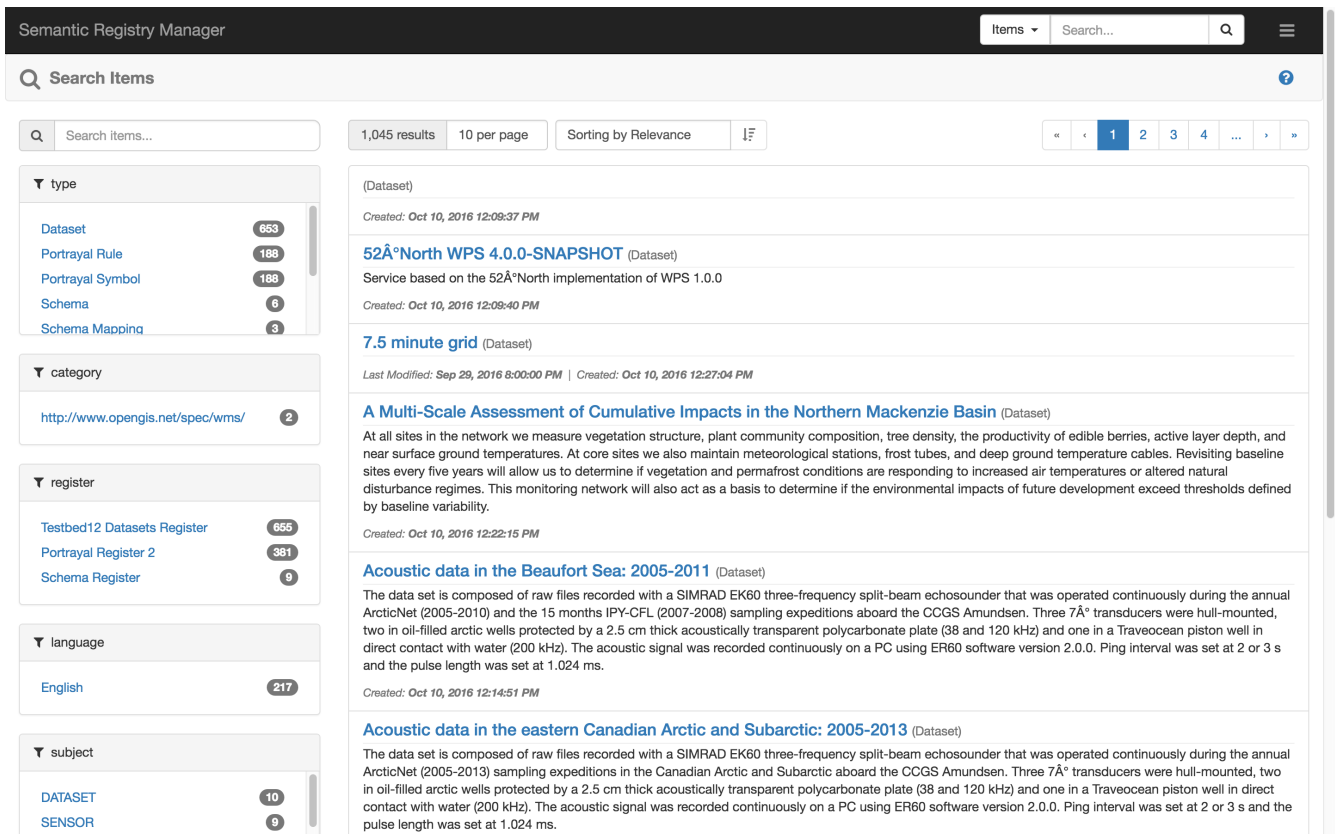


Figure 9. Image Matters Semantic Registry Client

## Integration with Semantic Mediation Service

The Semantic Mediation Service used the Semantic Registry to store and search for Schemas and Schema Mappings, so we considered the Semantic Mediation as a client since it delegates its CRUD and search operations to Semantic Registry. The following figure shows a schema mapping managed by the Semantic Registry:

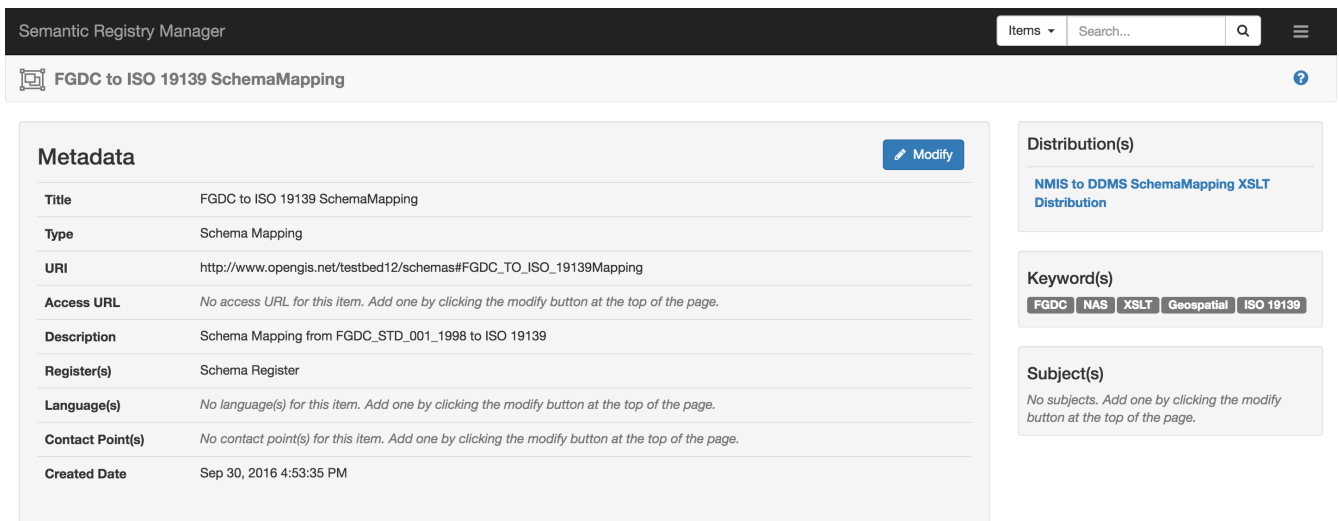
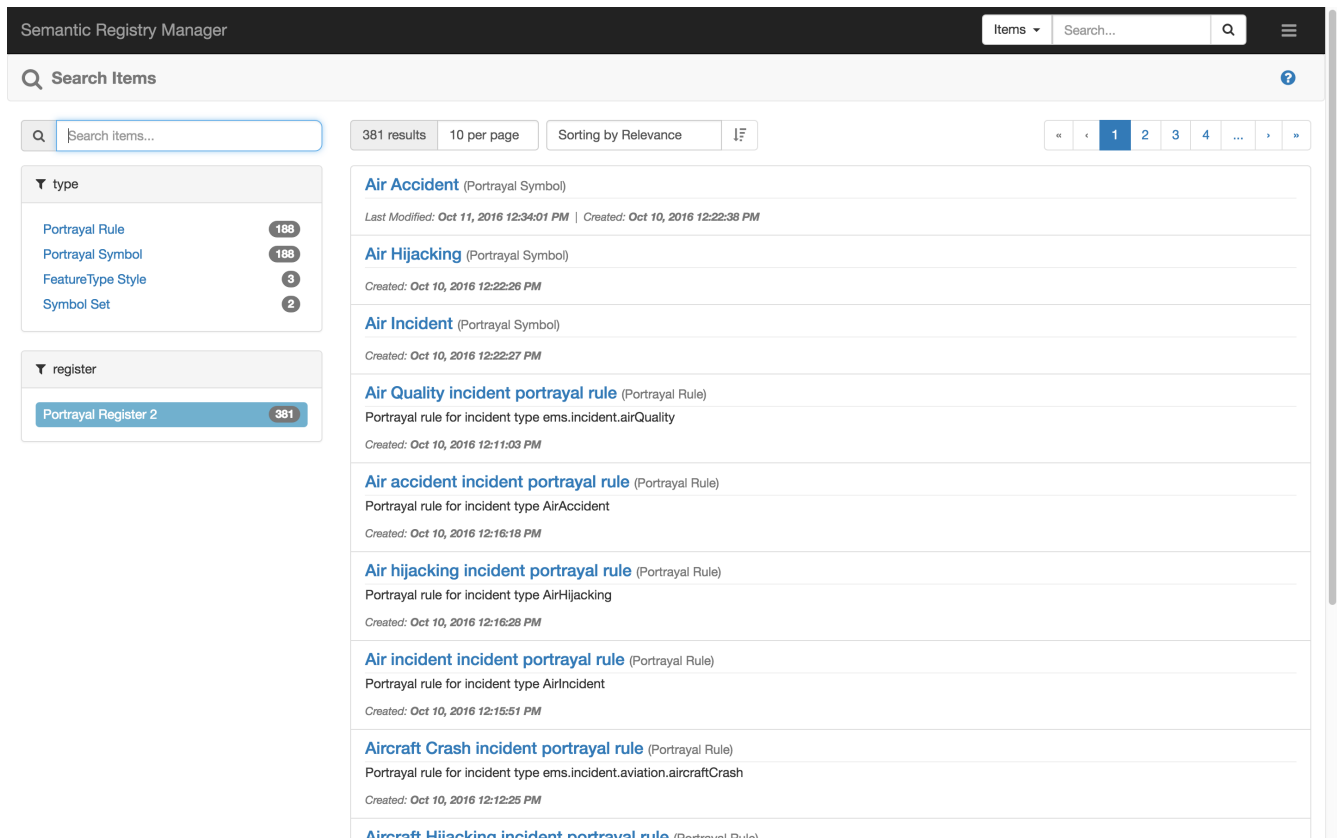


Figure 10. Schema Mapping Managed by Semantic Registry

## Integration with Semantic Portrayal Service

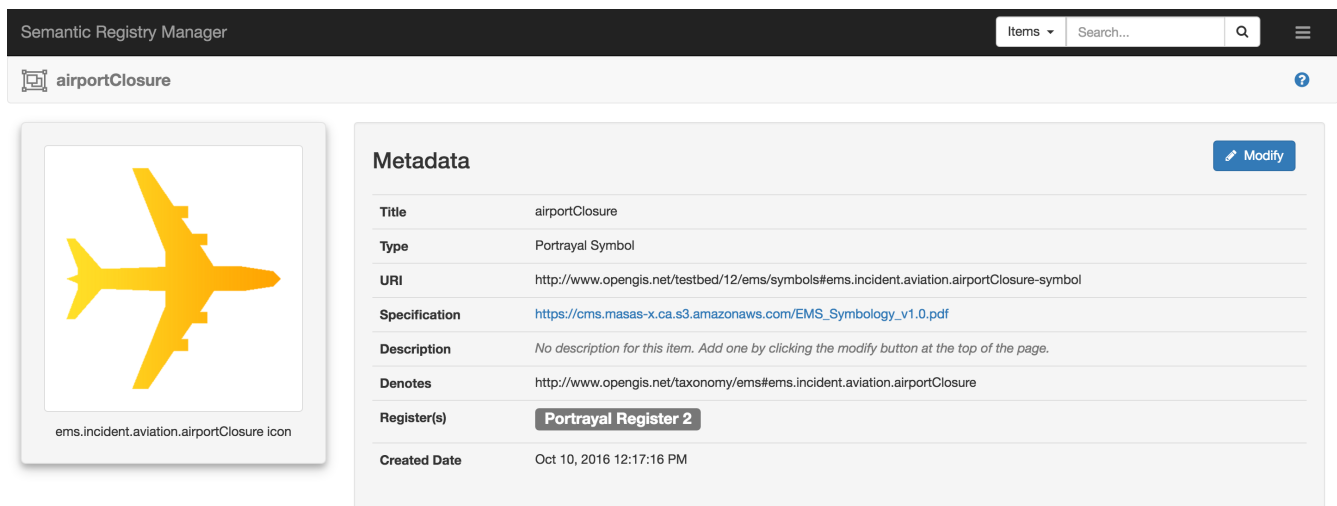
The Semantic Portrayal Service used the Semantic Registry to store and search for Portrayal information such as symbols, symbolizers, portrayal rules and feature type styles. We considered the Semantic Portrayal as a client since it delegates its CRUD and search operations to Semantic Registry. The following figure shows a faceted search for the portrayal register:



The screenshot displays the Semantic Registry Manager interface. At the top, there is a search bar with 'Items' and 'Search...' dropdowns. Below the search bar, a search filter is set to 'Search Items...'. The main content area shows 381 results, sorted by Relevance, with 10 items per page. The results are faceted by 'type' and 'register'. The 'type' facet includes Portrayal Rule (188), Portrayal Symbol (188), FeatureType Style (3), and Symbol Set (2). The 'register' facet includes Portrayal Register 2 (381). The main list of results includes items like 'Air Accident (Portrayal Symbol)', 'Air Hijacking (Portrayal Symbol)', 'Air Incident (Portrayal Symbol)', 'Air Quality incident portrayal rule (Portrayal Rule)', 'Air accident incident portrayal rule (Portrayal Rule)', 'Air hijacking incident portrayal rule (Portrayal Rule)', 'Air incident incident portrayal rule (Portrayal Rule)', and 'Aircraft Crash incident portrayal rule (Portrayal Rule)'. Each item shows its title, type, and creation/modification dates.

Figure 11. Portrayal Items Faceted Search

The following registry client snapshot shows the details of a symbol stored in the portrayal register:



The screenshot displays the Semantic Registry Manager interface showing the details of a symbol named 'airportClosure'. The symbol is represented by a yellow airplane icon. The metadata section includes the following information:

Field	Value
Title	airportClosure
Type	Portrayal Symbol
URI	http://www.opengis.net/testbed/12/ems/symbols#ems.incident.aviation.airportClosure-symbol
Specification	https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf
Description	No description for this item. Add one by clicking the modify button at the top of the page.
Denotes	http://www.opengis.net/taxonomy/ems#ems.incident.aviation.airportClosure
Register(s)	Portrayal Register 2
Created Date	Oct 10, 2016 12:17:16 PM

Figure 12. Symbol View managed by Register

# Chapter 9. Semantic Mediation Service

## 9.1. Overview

During the OGC Testbed 11, Image Matters LLC developed the first iteration of the Semantic Mediation Service, demonstrating the transformation of Homeland Security Working Group (HSWG) Incident Ontology to the Canadian Emergency Management Symbology (EMS) Ontology, using a rule engine. The engine was based on the **Semantic Mediation Ontology** and **SPARQL Extension ontology** expressing rules and functions of transformations between two semantic models (expressed as ontologies). The transformation from a source ontology to target ontology is called **Alignment** in the Semantic Mediation Ontology.

For this testbed, the focus was to use a **Schema Registry** to store information about schemas and schema mappings and to perform transformation of data represented in different schemas. The emphasis for this testbed was to use XML schemas and XSLT transformation, while still being able to accommodate the results from the previous testbed. Another objective of this testbed was to design the Semantic Mediation Service REST API and integrate it with the Semantic Registry and CSW eBRIM profile for Schema Registry.

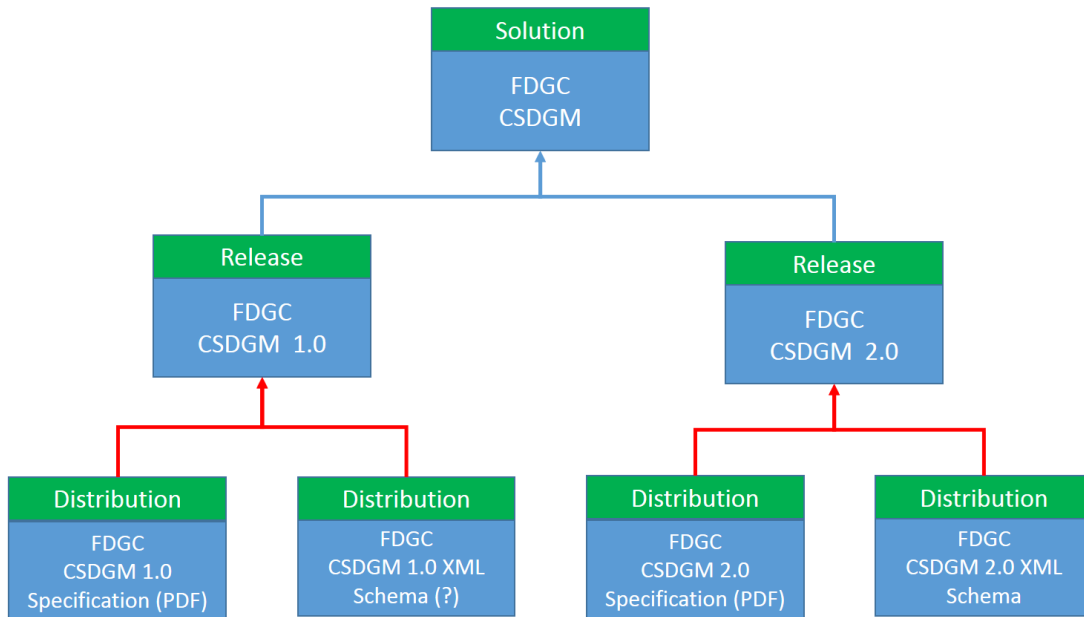
## 9.2. Analysis

A review of the existing standards that can represent schemas and schema mappings in a registry was performed. Only DCAT and ADMS were found relevant but no profile were defined to accommodate the specificities of schemas and schema mappings such as source and target schema in a mapping. A review of the eBRIM model in CSW was also performed and concluded that an extension was needed to represent schema and schema mapping information in the eBRIM model.

Very often schemas falls within the scope of a well-known standard name such as GML, WFS, FDGC CSGDM, NMIS, DDMS, etc. A given standard may have multiple **releases** (GML 1, GML 2, GML 3 for example). Each schema **release** within the standard is identified with a version or unique namespace. In the case of XML schema, the namespace is different for each version. However in the case of ontologies, the namespace is the same for each release of an ontology, however they often use owl:versionIRI to distinguish each release document. Each of this release can have multiple **distributions**: the schema document (XML Schema for example), its specification (reference to a PDF document for example). In the case of ontologies, a release may be composed of multiple encoding of the same ontology such as RDF, TTL, or N-TRIPLES. The following diagram shows how the FGDC CSDGM can be encoded using this three-layer approach.



# FGDC CSDGM Standard



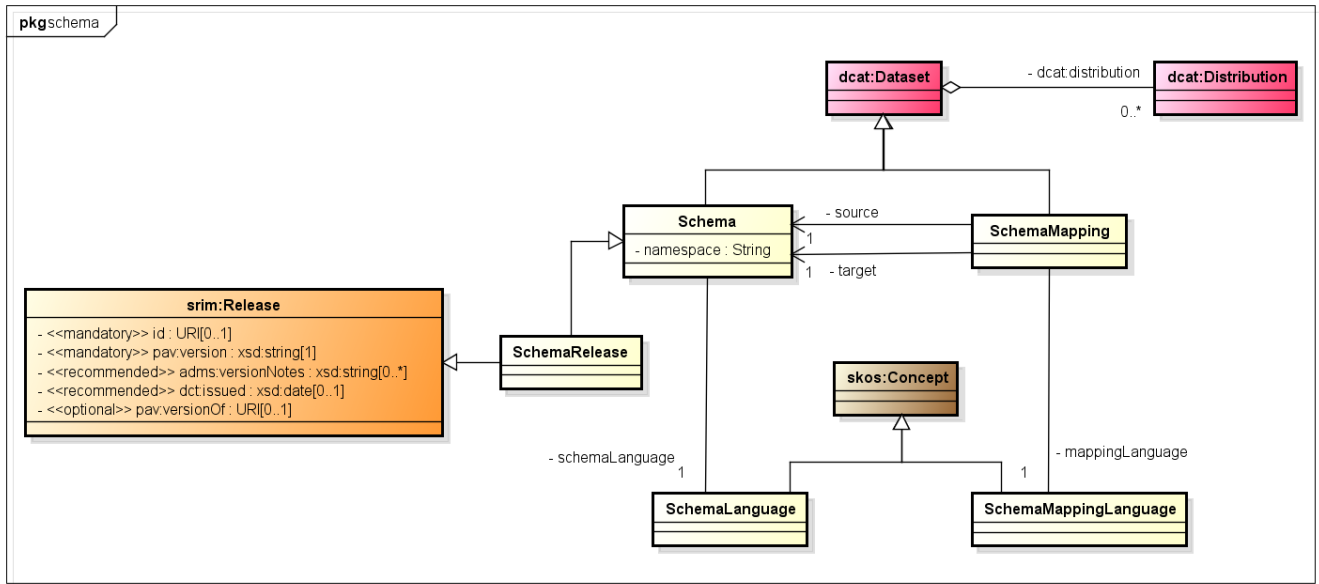
The DCAT model uses a two-layer approach (Dataset/Distribution). The SRIM Model introduces the concept of the **srim:Release** to accommodate the three-layer view to organization data information. The need of introducing the concept of **Release** was identified in the ADMS discussion forum, but no solution was proposed. We decided not to enforce one view over another, however we introduced the concept of **release** to enable users to better represent the organization of their information into three layers (Solution-Release-Distribution).

To enable the integration of Schema and Schema Mapping in the Semantic Registry, we developed during this testbed a SRIM profile for Schema and Schema Mapping. In addition, an eBRIM-Model extension was defined to register Schema and Schema Mapping in CSW eBRIM catalog. The following sections provides an overview of the models, REST API and integration work performed under this task. More details about the SRIM Application Profile and the Semantic Mediation REST API can be found in Appendix B and E.

## 9.3. SRIM Schema Application Profile

The SRIM Schema Application Profile is defined as an extension of the SRIM Core Model. A **Schema** and **SchemaMapping** have been defined a subclass of **dcam:Dataset**, thus inherits any of its properties when applicable. To enable a better search and discovery of schemas and schema mappings in a registry, we recommend to leverage the metadata recommended by the SRIM and DCAT models, and if possible organize the information into the three-layer approach describe in the previous section when applicable.

The overview of the model is shown in the figure below.



The following example shows how the NMIS v.2.2 XML Schema is encoded in Turtle Format.

```

@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .
@prefix adms: <http://www.w3.org/TR/vocab-adms/#> .
@prefix pav: <http://purl.org/pav/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix org: <http://www.w3.org/ns/org#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2>
  a schema:Schema ;
  dct:alternative "NGA.STND.0018_2.2" ;
  dct:description "An XML encoding of the NMIS logical model (Part 1)
that is conformant to ISO 19139 and the XML Data Encoding Specification for
Information Security Marking Metadata (DES.ISM.XML.V9)" ;
  dct:publisher <http://www.nga.mil> ;
  dct:title "National System for Geospatial Intelligence Metadata
Implementation Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2." ;
  pav:version "2.2" ; ①
  schema:namespace <http://metadata.ces.mil/dse/ns/GSIP/5.0/nas> ; ②
  schema:schemaLanguage <http://www.w3.org/TR/xmlschema-1/> ; ③
  skos:altLabel "NGA.STND.0018_2.2" ;
  dcat:distribution
    [ a dcat:Distribution ; ④
      dct:description "NGA.STND.0018_2.2 XML schema encoding"
    ;
      dct:title "NGA.STND.0018_2.2 XML Schema" ;
      dcat:accessURL
<http://schemas.opengis.net/iso/19139/20070417/gmd/gmd.xsd> ; ⑤
      dcat:mediaType "application/xml" ⑥
      adms:representationTechnique
<http://purl.org/adms/representationtechnique/XMLSchema> ⑦
    ] ;
  dcat:keyword "Metadata" , "Geospatial" , "NMIS" , "NGA" , "ISO
19139 Profile" , "XML Schema" .

<http://www.nga.mil>
  a org:Organization ;
  foaf:name "National Geospatial Intelligence Agency (NGA)" .

```

- ① **pav:version** to denote the version of the schema
- ② **schema:namespace** refers to the namespace URI associated with the schema
- ③ **schema:schemaLanguage** refers to a URL uniquely defining the Schema Language used to define the Schema encoding.
- ④ **dcat:Distribution** is used to define information to access the Schema
- ⑤ **dcat:accessURL** refers to the endpoint to download the schema
- ⑥ The **dct:mediaType** refers to the mime type of the XML Schema document (application/xml)

⑦ To distinguish which "flavor" of XML is used, we use the property **adms:representationTechnique** to refer to the XML Schema url standard.

The following schema shows the definition of the Department of Defense Discovery Metadata Specification (DDMS), Version 2.0".

```
@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .
@prefix adms: <http://www.w3.org/TR/vocab-adms/#> .
@prefix pav: <http://purl.org/pav/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix org: <http://www.w3.org/ns/org#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://metadata.dod.mil/mdr/ns/DDMS/2.0>
  a schema:Schema ;
  dct:description "Defines discovery metadata elements for resources
posted to community and organizational shared spaces" ;
  dct:publisher <http://www.nga.mil> ;
  dct:title "Department of Defense Discovery Metadata Specification
(DDMS), Version 2.0" ;
  pav:version "2012-07-13" ;
  schema:namespace <http://metadata.dod.mil/mdr/ns/DDMS/2.0/> ;
  schema:schemaLanguage <http://www.w3.org/TR/xmlschema-1/> ;
  dcat:distribution [ a dcat:Distribution ;
dct:description "" ;
dct:title "DDMS Schema file
(version 2012-07-13)" ;
adms:representationTechnique
<http://purl.org/adms/representationtechnique/XMLSchema> ;
dcat:accessURL
<http://schemas.opengis.net/iso/19139/20070417/gmd/gmd.xsd> ;
dcat:mediaType "application/xml"
] ;
  dcat:keyword "Metadata" , "Geospatial" , "DDMS" , "XML Schema" .

<http://www.nga.mil>
  a org:Organization ;
  foaf:name "National Geospatial Intelligence Agency (NGA)" .
```

The following example illustrates a schema mapping between NMIS 2.2 and DDMS 2.0 is defined in Turtle format.

```

@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .
@prefix adms: <http://www.w3.org/TR/vocab-adms/#> .
@prefix pav: <http://purl.org/pav/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix org: <http://www.w3.org/ns/org#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://www.opengis.net/testbed12/schemas#MNIS2DDMS>
  a schema:SchemaMapping ;
  dct:description "Schema Mapping from NMIS 2.2 to DDMS" ;
  dct:publisher <http://www.nga.mil> ;
  dct:title "NMIS 2.2 to DDMS SchemaMapping" ;
  pav:version "2013-12-2" ; ①
  schema:mappingLanguage <https://www.w3.org/TR/xslt20/> ; ②
  schema:sourceSchema
<http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2> ; ③
  schema:targetSchema <http://metadata.dod.mil/mdr/ns/DDMS/2.0> ; ④
  dcat:distribution [ a dcat:Distribution ;
⑤
                    dct:description "NGA.STND.0018_2.2 to
DDMS SchemaMapping XSLT Distribution" ;
                    dct:title "NMIS to DDMS
SchemaMapping XSLT Distribution" ;
                    dcat:accessURL
<http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&service=CSW-
ebRIM&id=urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4> ; ⑥
                    dcat:mediaType "application/xml"; ⑦
                    adms:representationTechnique
<http://purl.org/adms/representationtechnique/XSLT20> ; ⑧
                    ] ;
  dcat:keyword "XSLT" , "Geospatial" , "NMIS" , "DDMS" , "NAS" .

<http://www.nga.mil>
  a org:Organization ;
  foaf:name "National Geospatial Intelligence Agency (NGA)" .

```

- ① **pav:version** to denote the version of the schema mapping
- ② **schema:mappingLanguage** refers to a URL uniquely defining the Schema Mapping Language used to define the Schema Mapping encoding (XSLT in this case)
- ③ **schema:sourceSchema** refers to the source schema that this schema mapping accepts (NMIS v2.2 schema defined above).
- ④ **schema:targetSchema** refers to the target schema that this schema mapping maps to (DDMS v2.0 schema defined above).
- ⑤ **dcat:Distribution** is used to define information to access the XSLT document
- ⑥ **dcat:accessURL** refers to the endpoint to download the XSLT document

- ⑦ The **dc:mediaType** refers to the mime type of the XSLT document (application/xml)
- ⑧ To distinguish which "flavor" of XML is used, we use the property **adms:representationTechnique** to refer to the XSLT url standard.

During the testbed, we identified the need to standardize the taxonomies for Schema Language (RIF, RELAX-NG, EXPRESS, Avro Schema, JSON Schema, OWL, RDFS,...), Schema Mapping (RIF, XSLT, Programming language,...) and representation techniques (XSLT, RIF, XML Schema,...). ADMS provides a taxonomy for Representation techniques on the concept scheme: <http://purl.org/adms/representationtechnique/>. The top concepts of the taxonomy are shown below in RDF/XML format (note that it is missing XSLT, which we added in our example (<<http://purl.org/adms/representationtechnique/XSLT20>>))

```

<!-- http://purl.org/adms/representationtechnique/1.0 -->
<owl:NamedIndividual rdf:about="http://purl.org/adms/representationtechnique/1.0">
  <rdf:type rdf:resource="skos:ConceptScheme"/>
  <rdfs:label xml:lang="en">Representation Technique</rdfs:label>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/Archimate"/>
    <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/BPMN"/>
      <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/CommonLogic"/>
        <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/DTM"/>
          <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/DataLog"/>
            <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/Diagram"/>
              <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/GenericCode"/>
                <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/HumanLanguage"/>
                  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/IDEF"/>
                    <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/KIF"/>
                      <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/OWL"/>
                        <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/Prolog"/>
                          <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/RDFSchema"/>
                            <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/RIF"/>
                              <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/RelaxNG"/>
                                <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/RuleML"/>
                                  <skos:hasTopConcept

```

```

rdf:resource="http://purl.org/adms/representationtechnique/SBVR"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/SKOS"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/SPARQL"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/SPIN"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/SWRL"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/Schematron"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/TopicMaps"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/UML"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/WSDL"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/WSMO"/>
  <skos:hasTopConcept
rdf:resource="http://purl.org/adms/representationtechnique/XMLSchema"/>
  </owl:NamedIndividual>

```

The DCAT model allows multiple **dcate:Distribution** for a **dcate:Dataset**. Thus it is possible that a **schema:Schema** and **schema:SchemaMapping** (which are subclasses of **dcate:Dataset**) to have multiple distributions of the same schema or schema mapping using different representation techniques (for example a schema can be represented in RelaxNG or XML Schema). The use of **adms:representationTechnique** helps clients to select the distribution that they can support. This makes the **schema:schemaLanguage** and is less useful. On the other hand, having one distribution per Schema and SchemaMapping will make the use **schema:schemaLanguage** and **schema:mappingLanguage** more useful for search. There is however a thin line to distinguish the semantic of **schema:schemaLanguage** and **schema:mappingLanguage** with **adms:representationTechnique**.

To facilitate the integration with web clients that uses JSON, we defined a JSON-LD context to map **isomorphically** the RDF model to JSON-LD. This means that a JSON-LD document can be transformed to RDF model using the JSON-LD context and vice-versa. This explains the reason we use URI reference extensively in our JSON-LD representation, instead of internal value of particular system (ideally these URIs should be resolvable). This is default encoding used by the semantic registry and semantic mediation service.

The following example is the equivalent encoding of the NMISSchema in JSON-LD

```

{
  "id": "66dcd67e36e93540bc77e4710fd749ad",
  "uri": "http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2",
  "type": "schema:Schema",
  "register": [
    "schemas"
  ],
  "title": "National System for Geospatial Intelligence Metadata Implementation Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2.",
  "description": "An XML encoding of the NMIS logical model (Part 1) that is conformant to ISO 19139 and the XML Data Encoding Specification for Information Security Marking Metadata (DES.ISM.XML.V9)",
  "alternative": "NGA.STND.0018_2.2",
  "created": "2016-10-10T16:09:58.704Z",
  "publisher": [{
    "name": "National Geospatial Intelligence Agency (NGA)",
    "uri": "http://www.nga.mil",
    "type": "org:Organization"
  }],
  "keyword": [
    "ISO 19139 Profile",
    "XML Schema",
    "Metadata",
    "NGA",
    "Geospatial",
    "NMIS"
  ],
  "namespace": "http://metadata.ces.mil/dse/ns/GSIP/5.0/nas",
  "schemaLanguage": "http://www.w3.org/TR/xmlschema-1/",
  "distribution": [{
    "title": "NGA.STND.0018_2.2 XML Schema",
    "description": "NGA.STND.0018_2.2 XML schema encoding",
    "accessURL": "http://schemas.opengis.net/iso/19139/20070417/gmd/gmd.xsd",
    "type": "dcat:Distribution",
    "representationTechnique":
"http://purl.org/adms/representationtechnique/XMLSchema"
  }]
}

```

The following example is the equivalent encoding of the NMISSchema in JSON-LD



```

{
  "id": "2ddf2633f660b1d6f281a989faeec3e6",
  "uri": "http://www.opengis.net/testbed12/schemas#MNIS2DDMS",
  "type": "schema:SchemaMapping",
  "register": [
    "schemas"
  ],
  "title": "NMIS 2.2 to DDMS SchemaMapping",
  "description": "Schema Mapping from NMIS 2.2 to DDMS",
  "created": "2016-10-10T16:10:06.490Z",
  "publisher": [{
    "name": "National Geospatial Intelligence Agency (NGA)",
    "uri": "http://www.nga.mil",
    "type": "org:Organization"
  }],
  "keyword": [
    "NAS",
    "DDMS",
    "XSLT",
    "Geospatial",
    "NMIS"
  ],
  "sourceSchema": "http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2",
  "targetSchema": "http://metadata.dod.mil/mdr/ns/DDMS/2.0",
  "mappingLanguage": "https://www.w3.org/TR/xslt20/",
  "distribution": [{
    "title": "NMIS to DDMS SchemaMapping XSLT Distribution",
    "description": "NGA.STND.0018_2.2 to DDMS SchemaMapping XSLT Distribution",
    "accessURL":
"http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&service=CSW-
ebRIM&id=urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4",
    "type": "dcat:Distribution",
    "representationTechnique":
"http://purl.org/adms/representationtechnique/XSLT20"
  }]
}

```

The full description of the SRIM Schema Application Profile is described in the Appendix [\[SRIMSchemaProfile\]](#)

## 9.4. Schema Registry CSW-ebRIM profile

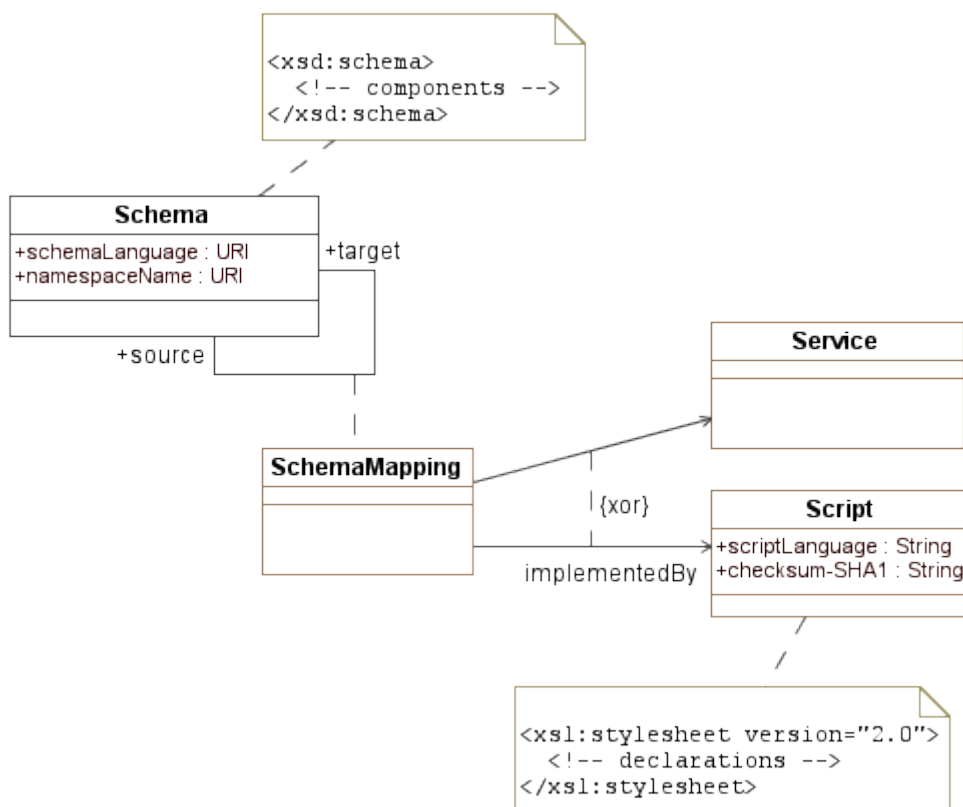
The CSW-ebRIM schema registry provided by Galdos is based on the CSW-ebRIM profile (OGC 07-110r4, OGC 07-144r4), with the following extensions:

- **Schema** (object type): A formal description of a model expressed using some schema language (e.g. XML Schema, RELAX NG, EXPRESS)
- **Script** (object type): An executable sequence of statements written in some scripting language

(e.g. XSLT, Python, shell script); it is usually interpreted rather than compiled.

- **SchemaMapping** (association type): A relation between two Schema objects that specifies how data structured under the source schema are to be transformed into data structured under the target schema.
- **ImplementedBy** (association type): A relation between a SchemaMapping and a Script or Service that implements the mapping in some executable way.
- **Ontology** (object type): A formal description of interrelated concepts and their properties that apply to a particular domain of discourse.

Two schemas may be related by a **SchemaMapping** association, which may be implemented by some script resource (or perhaps by a processing service). The following diagram illustrates the essential concepts.



The Schema and Script resources are shown above with repository items managed by the registry. It is also possible to register resources available elsewhere and link to an external representation using an **ExternalLink**.

## 9.5. Implementations

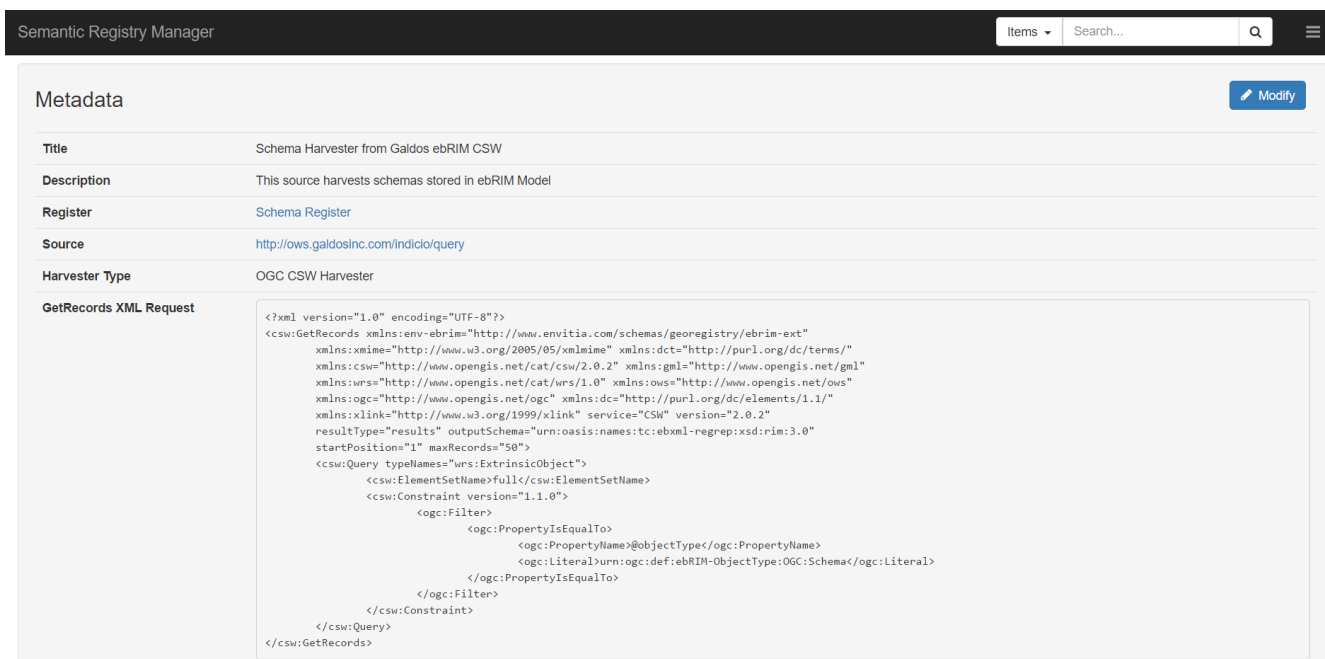
### 9.5.1. Integration of Galdos Schema Registry with Semantic Registry

To perform the integration of the Galdos Schema Registry with the Semantic Registry, two harvester types were defined in the Semantic Registry:

- **Schema Harvester**: harvest schema from CSW-ebRIM schema profile to SRIM Schema Profile.
- **Schema Mapping Harvester**: harvest schema mappings from CSW-ebRIM schema profile to

## SRIM Schema Profile.

Both harvester types were used to defined two harvester sources to the Galdos Schema Registry instance to convert the ebRIM profile to its semantic representation in Semantic Registry (thus can be query by its REST API or SPARQL endpoint). The following figure shows the configuration of the Schema Harvester in the Semantic Registry Client.



The screenshot shows the 'Semantic Registry Manager' interface. At the top, there is a search bar and a menu icon. Below that, the 'Metadata' section is displayed with a 'Modify' button. The metadata table contains the following information:

Title	Schema Harvester from Galdos ebRIM CSW
Description	This source harvests schemas stored in ebRIM Model
Register	Schema Register
Source	<a href="http://ows.galdosinc.com/Indicio/query">http://ows.galdosinc.com/Indicio/query</a>
Harvester Type	OGC CSW Harvester
GetRecords XML Request	<pre>&lt;?xml version="1.0" encoding="UTF-8"?&gt; &lt;csw:GetRecords xmlns:env-ebRim="http://www.envitia.com/schemas/georegistry/ebRim-ext"   xmlns:xmime="http://www.w3.org/2005/05/xmime" xmlns:dct="http://purl.org/dc/terms/"   xmlns:csw="http://www.opengis.net/cat/csw/2.0.2" xmlns:gml="http://www.opengis.net/gml"   xmlns:wrs="http://www.opengis.net/cat/wrs/1.0" xmlns:ows="http://www.opengis.net/ows"   xmlns:ogc="http://www.opengis.net/ogc" xmlns:dc="http://purl.org/dc/elements/1.1/"   xmlns:xlink="http://www.w3.org/1999/xlink" service="CSW" version="2.0.2"   resultType="results" outputSchema="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"   startPosition="1" maxRecords="50"&gt;   &lt;csw:Query typeNames="wrs:ExtrinsicObject"&gt;     &lt;csw:ElementSetName&gt;full&lt;/csw:ElementSetName&gt;     &lt;csw:Constraint version="1.1.0"&gt;       &lt;ogc:Filter&gt;         &lt;ogc:PropertyIsEqualTo&gt;           &lt;ogc:PropertyName&gt;@objectType&lt;/ogc:PropertyName&gt;           &lt;ogc:Literal&gt;urn:ogc:def:ebRIM-ObjectType:OGC:Schema&lt;/ogc:Literal&gt;         &lt;/ogc:PropertyIsEqualTo&gt;       &lt;/ogc:Filter&gt;     &lt;/csw:Constraint&gt;   &lt;/csw:Query&gt; &lt;/csw:GetRecords&gt;</pre>

Figure 13. Schema Harvester Source for ebRIM CSW

The request filter constrains the namespace names of both the source and target schemas. The response includes a tuple (conveyed by the rim:RegistryObjectList element) that contains the matching schema mapping and an implementation (a Script or a Service object). In this case it's a Script (media type: "application/xslt+xml") resource that can be downloaded from the registry using the URI given by wrs:repositoryItemRef/@xlink:href (a link to the repository item).

```
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
  xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
  xmlns:ogc="http://www.opengis.net/ogc"
  service="CSW"
  version="2.0.2"
  maxRecords="10"
  resultType="results">
  <Query typeNames="rim:Association_a1_a2 wrs:ExtrinsicObject_e1_e2
rim:RegistryObject_r1">
    <ElementSetName typeNames="a1 r1">full</ElementSetName>
    <Constraint version="1.1.0">
      <ogc:Filter>
        <ogc:And>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>$a1/@associationType</ogc:PropertyName>
            <ogc:Literal>urn:ogc:def:ebRIM-
AssociationType:OGC:SchemaMapping</ogc:Literal>
```

```

</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
  <ogc:PropertyName>$a1/@sourceObject</ogc:PropertyName>
  <ogc:PropertyName>$e1/@id</ogc:PropertyName>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
  <ogc:PropertyName>$e1/@objectType</ogc:PropertyName>
  <ogc:Literal>urn:ogc:def:ebRIM-ObjectType:OGC:Schema</ogc:Literal>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
<ogc:PropertyName>$e1/rim:Slot[@name='http://www.opengis.net/cat/wrs/1.0/namespaceName
']/rim:ValueList/rim:Value</ogc:PropertyName>
  <ogc:Literal>http://metadata.ces.mil/dse/ns/GSIP/5.0/nas</ogc:Literal>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
  <ogc:PropertyName>$a1/@targetObject</ogc:PropertyName>
  <ogc:PropertyName>$e2/@id</ogc:PropertyName>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
  <ogc:PropertyName>$e2/@objectType</ogc:PropertyName>
  <ogc:Literal>urn:ogc:def:ebRIM-ObjectType:OGC:Schema</ogc:Literal>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
<ogc:PropertyName>$e2/rim:Slot[@name='http://www.opengis.net/cat/wrs/1.0/namespaceName
']/rim:ValueList/rim:Value</ogc:PropertyName>
  <ogc:Literal>http://metadata.dod.mil/mdr/ns/DDMS/2.0/</ogc:Literal>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
  <ogc:PropertyName>$a2/@associationType</ogc:PropertyName>
  <ogc:Literal>urn:ogc:def:ebRIM-
AssociationType:OGC:ImplementedBy</ogc:Literal>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
  <ogc:PropertyName>$a2/@sourceObject</ogc:PropertyName>
  <ogc:PropertyName>$a1/@id</ogc:PropertyName>
</ogc:PropertyIsEqualTo>
<ogc:PropertyIsEqualTo>
  <ogc:PropertyName>$a2/@targetObject</ogc:PropertyName>
  <ogc:PropertyName>$r1/@id</ogc:PropertyName>
</ogc:PropertyIsEqualTo>
</ogc:And>
</ogc:Filter>
</Constraint>
</Query>
</GetRecords>

```

```

<GetRecordsResponse xmlns='http://www.opengis.net/cat/csw/2.0.2' version='2.0.2'>
  <SearchStatus timestamp='2016-10-13T16:23:04Z'></SearchStatus>

```

```

<SearchResults numberOfRecordsMatched='1' elementSet='full'
numberOfRecordsReturned='1' nextRecord='0'>
  <RegistryObjectList xmlns='urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0'>
    <rim:Association xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:xlink="http://www.w3.org/1999/xlink" id="urn:uuid:2c14d651-dc53-4367-91bc-62ca7db6f40c" lid="urn:uuid:2c14d651-dc53-4367-91bc-62ca7db6f40c"
objectType="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Association"
status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
associationType="urn:ogc:def:ebRIM-AssociationType:OGC:SchemaMapping"
sourceObject="urn:uuid:566ff231-e768-41de-bb26-e0def91ceb9b"
targetObject="urn:uuid:02200156-4271-4d52-a2e1-47d268680aa7">
      <rim:VersionInfo versionName="20160902T192540Z"/>
    </rim:Association>
    <wrs:ExtrinsicObject xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:xlink="http://www.w3.org/1999/xlink" id="urn:uuid:38ffef44-e1ee-4856-8812-
d8b82da497a4" lid="urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4"
objectType="urn:ogc:def:ebRIM-ObjectType:OGC:Script" status="urn:oasis:names:tc:ebxml-
regrep:StatusType:Submitted" mimeType="application/xslt+xml">
      <rim:Slot name="http://www.opengis.net/cat/wrs/1.0/scriptLanguage">
        <rim:ValueList>
          <rim:Value>https://www.w3.org/TR/xslt20/</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="http://purl.org/dc/terms/date"
slotType="urn:oasis:names:tc:ebxml-regrep:DataType:Date">
        <rim:ValueList>
          <rim:Value>2016-05-11</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Slot name="http://purl.org/dc/terms/extent">
        <rim:ValueList>
          <rim:Value>59 kB</rim:Value>
        </rim:ValueList>
      </rim:Slot>
      <rim:Name>
        <rim:LocalizedString xml:lang="en" charset="UTF-8"
value="NMFtoDDMS2-20160511.xsl"/>
      </rim:Name>
      <rim:Description>
        <rim:LocalizedString xml:lang="en" charset="UTF-8" value="XSLT v2
stylesheet that accepts an NMIS 2.2 document as input and generates a DDMS 2.0
compliant document as output."/>
      </rim:Description>
      <rim:VersionInfo versionName="20160902T192540Z"/>
      <rim:ContentVersionInfo versionName="20160902T192540Z"/>
      <wrs:repositoryItemRef xlink:type="simple"
xlink:href="http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&se

```

```

rvice=CSW-ebRIM&id=urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4"/>
    </wrs:ExtrinsicObject>
  </RegistryObjectList>
</SearchResults>
</GetRecordsResponse>

```

The conversion to the JSON-LD representation of the SchemaMapping is represented below.

```

{
  "id": "1854365d6a9ea7a88788e3380ff4d770",
  "uri": "urn:uuid:2c14d651-dc53-4367-91bc-62ca7db6f40c",
  "type": "schema:SchemaMapping",
  "register": [
    "schemas"
  ],
  "created": "2016-10-10T16:21:20.237Z",
  "sourceSchema": "urn:uuid:566ff231-e768-41de-bb26-e0def91ceb9b",
  "targetSchema": "urn:uuid:02200156-4271-4d52-a2e1-47d268680aa7",
  "distribution": [{
    "uri": "urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4",
    "type": "dcat:Distribution",
    "title": "NMFtoDDMS2-20160511.xsl",
    "titleMap": {
      "en": "NMFtoDDMS2-20160511.xsl"
    },
    "description": "XSLT v2 stylesheet that accepts an NMIS 2.2 document as input
and generates a DDMS 2.0 compliant document as output.",
    "descriptionMap": {
      "en": "XSLT v2 stylesheet that accepts an NMIS 2.2 document as input and
generates a DDMS 2.0 compliant document as output."
    },
    "representationTechnique": "https://www.w3.org/TR/xslt20/"
  }]
}

```

In addition to the harvesting of the ebRIM Schema Registry, we populated the registry with additional schemas with more details attributes using RDF encoding and JSON-LD representation (using the Semantic Registry REST API to post items).

### 9.5.2. Semantic Mediation Service

The Semantic Mediation Service was build as convenience REST API that uses the Semantic Registry to search and discover schemas and schema mappings, but provides additional functionalities such as schema validation and schema transformation.

The REST API of the Semantic Mediation Registry was designed on the same criteria than the Semantic Registry REST API. It supports Level 2 and Level 3 REST API. The Level 3 is implemented with HAL. The payload of the REST API is using JSON that can be converted to Linked Data using a JSON-LD context referred at the entry point of the service. The REST API provides endpoints to

search schemas and schema mappings, perform validation and transformation between two schemas. The detailed description of the REST API can be found in Appendix E.

# Chapter 10. Semantic Portrayal Service

## 10.1. Overview

The initial implementation of the Semantic Portrayal Service during the OGC Testbed 11 focused on defining the styles, portrayal rules, point-based symbols and graphics to enable a Web Processing Service (WPS) to produce a SLD document. The initial ontology was heavily based in ISO 19117 Geographic Information-Portrayal standard. However during of our implementation of style renderers and development of the graphic ontology, we concluded the ISO 19117 was mostly designed for runtime implementation (for example use of portrayal function) than adapted for a declarative approach. The OGC SE standard provides a declarative approach based on XML encoding that is better aligned with modern renderer API (Java Canvas, HTML Canvas, SVG, MapCSS, ESRI Map Renderer, etc.).

We try to take the best ideas for different standards to define our portrayal ontologies and modularize them in such as way that it favors reusability

## 10.2. Review of symbology graphic formats

A number of existing standards related to Portrayal were considered during this testbed to come out with the portrayal ontologies.

### 10.2.1. Symbol Encoding (SE)

This Specification 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.

### 10.2.2. OGC Style Layer Descriptors (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.

### 10.2.3. Scalable Vector Graphics (SVG)

SVG has been in development since 1999 by a group of companies within the W3C after the competing standards Precision Graphics Markup Language (PGML, developed from Adobe's



PostScript) and Vector Markup Language (VML, developed from Microsoft's RTF) were submitted to W3C in 1998. SVG drew on experience from the designs of both those formats. SVG allows three types of graphic objects: vector graphics, raster graphics, and text. Graphical objects, including PNG and JPEG raster images, can be grouped, styled, transformed, and composited into previously rendered objects.

#### 10.2.4. 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. It allows feature data to be separate from portrayal data, permitting data to be portrayed in a dataset independent manner.

#### 10.2.5. 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.

### 10.3. Portrayal Ontologies

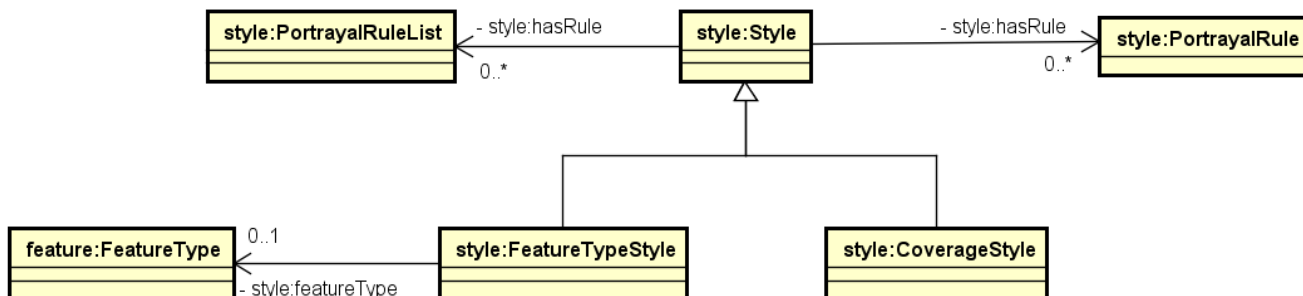
The Portrayal Ontologies specify a conceptual model for portrayal data, in particular symbols and portrayal rules. Portrayal rules associate features with symbols for the portrayal of the features on maps and other display media. These ontologies include classes, attributes and associations that provide 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, SVG, KML).

To favor reusability, the Portrayal ontologies are decomposed into four microtheories:

- **Style ontology:** defines the concept of Style and portrayal rules.
- **Symbol ontology:** defines the concept of SymbolSet and Symbol and structural definition of Symbol components.
- **Symbolizer ontology:** defines the concepts of Symbolizers defining instructions to render data as graphics.
- **Graphic Ontology:** defines graphic elements including graphic objects and attributes.

### 10.3.1. Style Microtheory

During this testbed, we extended the Style Microtheory designed during the testbed 11 to accommodate contextual information such as the scale range of application of a portrayal and better modeling schema used by the style rules by leveraging the work done for the semantic mediation and schema registry.



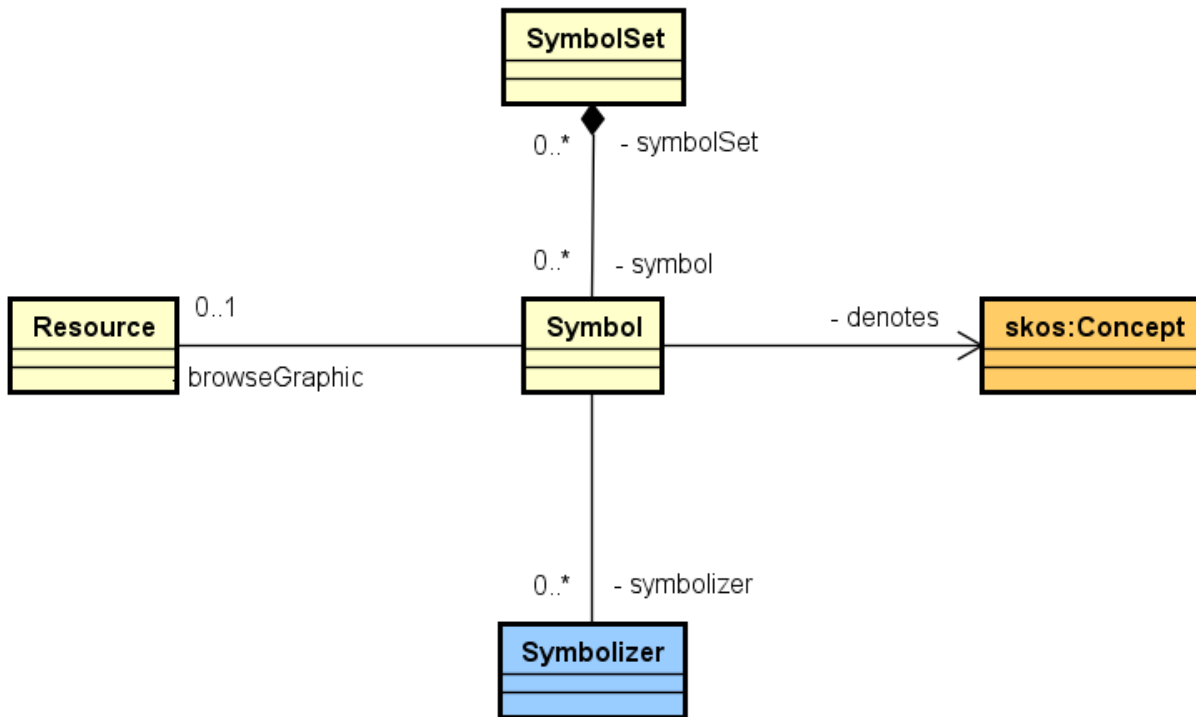
The previous Style ontology developed in Testbed 11 uses specialized properties (sparqlCondition, rifCondition, ogcFilterCondition). For this testbed, we modified the ontology to provide an extensible mechanism to express constraint in any language (which is more aligned with the CSW approach). We introduce the concept of **style:Constraint**. We also added introduce the minScaleDenominator and maxScaleDenominator as property of the **RuleCondition**, as they are common properties for style.

The details of the style microtheory are described in the Appendix C.

### 10.3.2. Symbol Microtheory

The Symbology Microtheory is an ontology that defines the conceptual model for defining **SymbolSet** and **Symbol**. A **Symbol** provides a graphic representation that denotes a concept (typically a feature type with specific properties). For this testbed, we extended the Symbol Microtheory designed during the testbed 11 to accommodate text, line, and area based symbol components and formalized binding between data properties (geometry) and graphic properties. The notion of Symbol is missing in the OGC SE standard as symbolizers are directly related to a FeatureTypeStyle. Introducing this intermediary Symbol Concept favors reusability of the symbols between different FeatureTypeStyle but also linking symbols to other concepts such as SymbolSet, concepts that they denote, allowing better search and mediation of symbols between different domains.

Due to the lack of time, we only address symbols that are not parameterizable (except their binding to geometry property). Future extension may address this requirement by introducing subclass of **Symbol**, such as **SymbolTemplate** (as introduced in ISO 19117).



### 10.3.3. Symbolizer Ontology

The Symbolizer ontology defines a set of symbolizers (also called renderers) that defines instructions for rendering data to graphic representation. The ontology is built on top of the Graphic Ontology which defines graphic objects and properties. Symbolizers are referred by **symbol:Symbol** to describe how conceptual symbol are rendered into graphics. The details of the ontology is described in Appendix C. This section describes an overview of the microtheory, examples and rationale of some of the design decisions.

The OGC Symbology Encoding defines a number of symbolizers that uses SVG parameters defines a key value pair. While this mechanism provides extensibility by accomodate future key value pairs, there is no schema allowing to know which keys are valid for a given symbolizer. We decided to move away from the approach used in the previous testbed 11, based on the ISO 19117 model which introduces symbol definition, symbol components and graphics. We found out that the ISO 19117 was not adequate to represented as semantic descriptive model. The ISO 19117 is more geared toward an implementation which can use function to calculate positioning of symbol components. We decided to align out model more toward main stream map renderer and descriptive standards such as SVG. === Symbolizer Hierarchy

The top concept of the Symbolizer ontology is **Symbolizer**. A **Symbolizer** describes how a feature is to appear on a map. The Symbolizer describes not just the shape that should appear but also how visual variables are defined using graphical properties such as color and opacity. A Symbolizer is obtained by specifying one of subclass of Symbolizer and then supplying parameters to override its default behavior. The hierarchy of symbolizer is illustrated in the following figure.

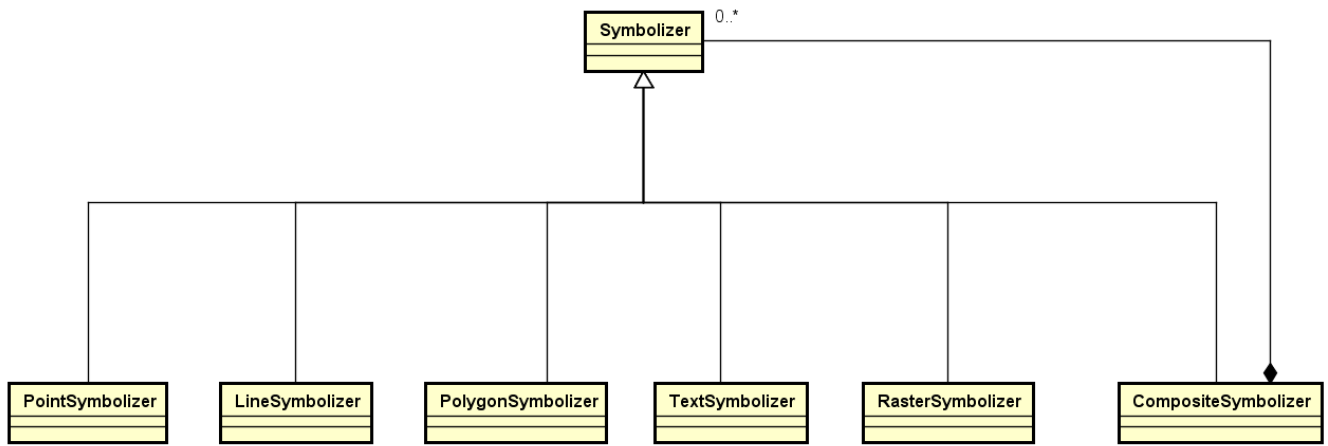


Figure 14. Symbolizer Hierarchy

Note that the hierarchy uses similar terminology than the OGC SE specification, except that it introduces an additional class called **CompositeSymbolizer** that defines a composition of symbolizers applied in a given order and using a composition operation (default is source over). This version of the ontology mostly focused on rendering of feature (vector) data, and is not addressing in depth the Raster Symbolizer, which is introduced as a placeholder for future extension. Other extensions of symbolizer may be defined down the road for more specialized use-cases such as complex symbologies that are difficult to express with graphic descriptors (example MIL 2525 symbology).

### Point Symbolizer

A **PointSymbolizer** is defined as a **Symbolizer** that is used to draw a “graphic” at a point. If the geometry associated with the **PointSymbolizer** is not a point, such as a line, polygon, raster, a representative point of the geometry (typically centroid) should be used.

### Line Symbolizer

A **LineSymbolizer** is a **Symbolizer** that is used to render a "stroke" along a linear geometry type such as string of line segments. Geometry types other than inherently linear types can also be used. If a point geometry is used, it should be interpreted as a line of “epsilon” (arbitrarily small) length with a horizontal orientation centered on the point, and should be rendered with two end caps. If a "area" geometry (for example polygon) is used, then its closed outline is used as the line string (with no end caps). If a raster geometry is used, its coverage-area outline is used for the line, rendered with no end caps.

### Polygon Symbolizer

A **PolygonSymbolizer** is a **Symbolizer** that is used to render the area enclosed by a polygon. If a polygon has “holes,” then they are not filled, but the borders around the holes are stroked in the usual way. “Islands” within holes are filled and stroked, and so on. If a point geometry is referenced instead of a polygon, then a small, square, ortho-normal polygon should be constructed for rendering. If a line is referenced, then the line (string) is closed for filling (only) by connecting its end point to its start point, any line crossings are corrected in some way, and only the original line is stroked. If a raster geometry is used, then the raster-coverage area is used as the polygon.

## Text Symbolizer

A **TextSymbolizer** is a **Symbolizer** that is used to render text.

## Raster Symbolizer

A **RasterSymbolizer** is a **Symbolizer** that is used to render raster. This symbolizer has been introduced as a placeholder for future extensions, as this testbed was primarily focused on symbolization of vector data.

## Composite Symbolizer

A **CompositeSymbolizer** is a symbolizer composed of a list of Symbolizers that applied in a given order using a given composition operation. Default composition operation is **src-over** composition.

## Property

A **Property** defines a reference to a property name or property path for a given type pointing to a value used for rendering. It can be a simple property name, a XPath (using slash separator) for XML Data Model, a json path (using dot notation) for JSON data model or a SPARQL path for Linked Data Model. The introduction of the class is intended to be a generalization of the `ogc:PropertyName` and `se:Geometry` to accomodate different data models (Linked Data, JSON, XML Model).

## Graphics Microtheory

The bulk of the effort for this thread was to flush out the Graphics Ontology used to describe the graphic elements of the symbols. The scope of the ontology is to be able to capture the graphic objects and properties expressed in different graphic encodings such as OGC Symbol Encoding (SE), SVG, KML. While the previous Testbed 11 was focused on Point symbology, and introduced `PointIcon` pointing to external graphics such as Font (HSWG incident symbols) and image-based symbols (Emergency Management Symbology), this effort extended the graphic ontology by addressing the following requirements:

- Point symbology
- Line symbology
- Area symbology
- Text symbology
- Graphic properties compatible with SVG
- Graphic Symbol defined by geometries and marks.
- Binding of graphic properties to data properties.
- Rendering to SVG and PNG

The rationale to introduce a Graphic Ontology is to provide a vocabulary describing graphic elements (objects and properties) and encodes them in an independent way. The use of ontology language such as OWL provide a powerful mechanism to express the semantic, classification, relationships and constraints of the graphic elements. The use of Linked data encoding for the graphics enable to build a "Web of Graphics" that could be referenced by other resources and the

decentralized distribution of graphics information on the web, favoring reusability of graphics and linkage to application that can go beyond styling application.

## Approach

Before starting the encoding of the Graphics Ontology, we perform an analysis of the different graphics standards related to the map symbology and graphic drawing in general. We identified the common graphic elements (objects and properties). The following standards were considered in the analysis:

- OGC Style Layer Description (SLD)
- OGC Symbology Encoding (SE)
- Scalable Vector Graphics (SVG)
- OGC Keyhole Markup Language (KML)
- ISO 19117 Geographic Information- Portrayal
- [CartoCSS](#)
- [MapCSS](#)
- [SVG Graphics Ontology](#)
- [Military Symbology](#)
- [Visualisation Ontology \(VISO\)](#)

Our analysis results find out that SVG and CSS graphic elements were the most commonly used among the different standards. SVG provides also the richest set of graphic elements that can accommodate a broad range of graphic drawing applications. We identified an initial list of graphic elements that are the most commonly used for map drawing and we formalized a new ontology as the existing graphic ontologies such as VISO or SVG Graphics Ontology were not adequate as they were too generic or over engineered for map portrayal.

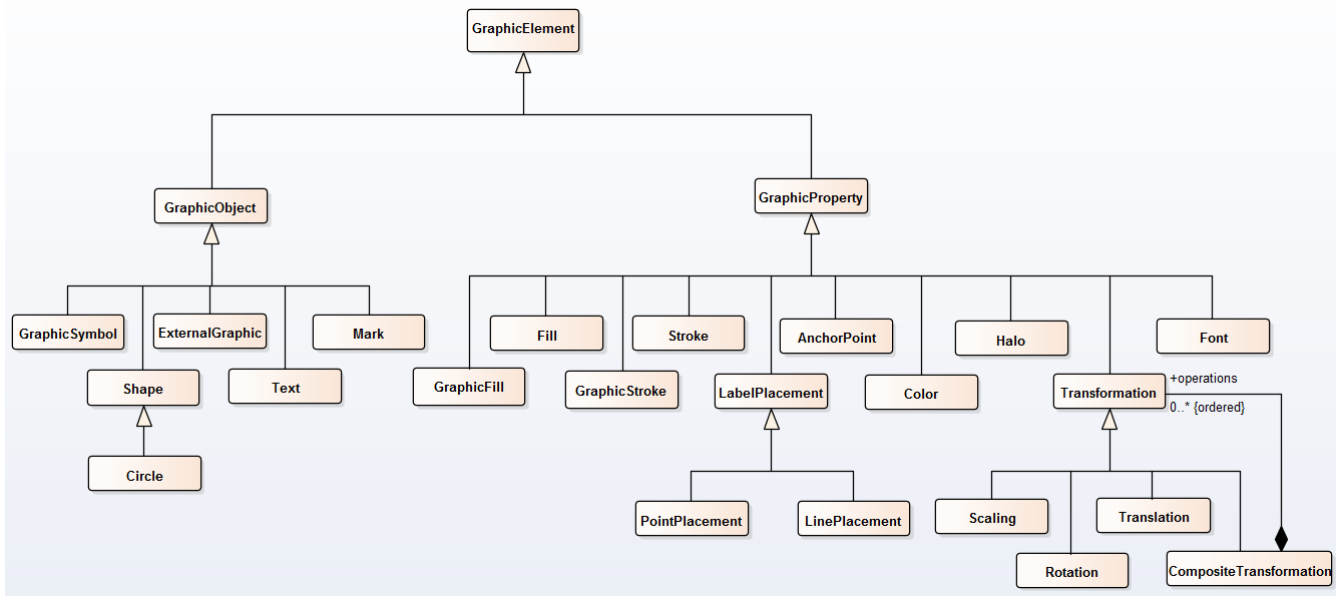
To model the graphic elements of the ontology, we try to use a model that is close to the way graphic properties are defined in CSS and SVG. For example to encode color, the preferred way is to use literal with a `CSSColorLiteral` datatype that uses the same syntax than CSS and SVG. Similarly length can be expressed using different unit of measures (percent, cm, mm). Instead of introducing a concept of Length and Unit of Measure that could be very verbose, we choose to use introduce a datatype `LengthType` that is based on the SVG and CSS syntax.

## Overview of the model

The **graphic:GraphicElement** concept is the top concept of the Graphic Ontology. It is aligned with the definition introduced in ISO 19117. The range of the **graphic:graphicContent** property of a **symbol:SymbolComponent** is a **graphic:GraphicElement**. A `GraphicElement` has two subclasses: **graphic:GraphicObject** and **graphic:GraphicProperty** (also aligned with the ISO 19117 model).

A **graphic:GraphicObject** is defined as a `graphic:GraphicElement` that can be drawn such as a graphic shape (oval, ellipse, rectangle, path) , Raster, Glyphs (text and graphic symbols) .

An overview of the Graphic Ontology hierarchy is shown below:



Graphic properties represents attributes that are used to modify the appearance of the graphic objects such as its size, orientation, color, outline stroke, fill pattern, fonts, font size. Several standards such as ISO 19117 and OGC SE encodes graphic properties as name/value pairs, which provide flexibility for future extensions with new names, but they do not provide any semantic to the name and make it hard to define the adequate restrictions of graphic properties on specific graphic objects. It is also prone to misinterpretation by implementers. The use of ontology provides a clear semantic for each property and relationships to other properties. For example, **graphic:font-color**, **graphic:stroke-color** and **graphic:fill-color** are defined as subproperties of **graphic:color**. The graphic ontology introduces two ways to model graphic properties. The first one is by defining subproperties of the datatype property **graphic:graphicProperty** (for example **graphic:stroke-width**, **graphic:stroke-color**, **graphic:font-size**). The second way is by defining a subclass of the **graphic:GraphicProperty** such as **graphic:Stroke**, **graphic:Color**, **graphic:Font**, **graphic:Fill**, which are often used as reusable containers for of multiple graphic properties. More details about the Graphic Ontology can be found in Appendix C.

### Semantic Portrayal Service REST API

The Semantic Portrayal Service implementation is accessible through a hypermedia-driven REST-based API to access style information (styles, ruleset, symbolsets, symbols) from the service.

- Hypermedia-driven REST API using Hypermedia Application Language JSON (HAL-JSON) (Level 3 on Richardson Maturity Model)
- JSON API (Level 2 on Richardson Maturity Model)
- Linked Data API.

The style information are encoded in following representations: RDF/XML, Turtle, N-Triples, JSON-LD and HAL-JSON. The Semantic Portrayal Service REST API is described in more details in Appendix F.

# Appendix A: Semantic Registry Information Model (SRIM)

In the following sections, classes and properties are grouped under headings ‘mandatory’, ‘recommended’ and ‘optional’. These terms have the following meaning.

- **Mandatory class:** a receiver of data **MUST** be able to process information about instances of the class; a sender of data **MUST** provide information about instances of the class.
- **Recommended class:** a sender of data **SHOULD** provide information about instances of the class; a sender of data **MUST** provide information about instances of the class, if such information is available; a receiver of data **MUST** be able to process information about instances of the class.
- **Optional class:** a receiver **MUST** be able to process information about instances of the class; a sender **MAY** provide the information but is not obliged to do so.
- **Mandatory property:** a receiver **MUST** be able to process the information for that property; a sender **MUST** provide the information for that property.
- **Recommended property:** a receiver **MUST** be able to process the information for that property; a sender **SHOULD** provide the information for that property if it is available.
- **Optional property:** a receiver **MUST** be able to process the information for that property; a sender **MAY** provide the information for that property but is not obliged to do so.

The meaning of the terms **MUST**, **MUST NOT**, **SHOULD** and **MAY** in this section and in the following sections are as defined in RFC 2119.

In the given context, the term "processing" means that receivers must accept incoming data and transparently provide these data to applications and services. It does neither imply nor prescribe what applications and services finally do with the data (parse, convert, store, make searchable, display to users, etc.).

Classes are classified as **Mandatory** if they appear as the range of one of the mandatory properties.

## Namespaces

Table 2. Namespaces

Prefix	Namespace URI	Schema & Documentation
adms	<a href="http://w3.org/ns/adms">http://w3.org/ns/adms</a>	<a href="#">Asset Description Metadata Schema (ADMS)</a>
dcat	<a href="http://www.w3.org/ns/dcat#">http://www.w3.org/ns/dcat#</a>	<a href="#">Data Catalog Vocabulary</a>
dct	<a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a>	<a href="#">DCMI Metadata Terms</a>
extent	<a href="http://www.opengis.net/ont/spatial/extent">http://www.opengis.net/ont/spatial/extent</a>	N/A



foaf	<a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a>	FOAF Vocabulary
geosparql	<a href="http://www.opengis.net/ont/geosparql">http://www.opengis.net/ont/geosparql</a>	GeoSPARQL
gr	<a href="http://www.heppnetz.de/ontologies/goodrelations/v1#">http://www.heppnetz.de/ontologies/goodrelations/v1#</a>	GoodRelations Specification
id	<a href="http://www.knowledgesmarts.com/ontologies/identifier#">http://www.knowledgesmarts.com/ontologies/identifier#</a>	N/A
link	<a href="http://www.opengis.net/ont/link#">http://www.opengis.net/ont/link#</a>	N/A
locn	<a href="http://www.w3.org/ns/locn#">http://www.w3.org/ns/locn#</a>	ISA Programme Core Location Vocabulary
org	<a href="http://www.socialml.org/ontologies/organization#">http://www.socialml.org/ontologies/organization#</a>	Organization Vocabulary
prov	<a href="http://www.w3.org/ns/prov#">http://www.w3.org/ns/prov#</a>	PROV-O: The PROV Ontology
rdf	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>	Resource Description Framework (RDF): Concepts and Abstract Syntax
rdfs	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>	RDF Vocabulary Description Language 1.0: RDF Schema
schema	<a href="http://schema.org/">http://schema.org/</a>	Schema Vocabulary
skos	<a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a>	SKOS Simple Knowledge Organization System - Reference
srin	<a href="http://www.opengis.net/ont/testbed12/srin#">http://www.opengis.net/ont/testbed12/srin#</a>	N/A
vcard	<a href="http://www.w3.org/2006/vcard/ns#">http://www.w3.org/2006/vcard/ns#</a>	vCard Ontology
xsd	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>	XML Schema Part 2: Datatypes Second Edition

## Registry Core Ontology

### Overview

To facilitate the integration of geospatial assets (such as datasets, services, portrayal information, schemas, maps, layers), semantic metadata plays a central role in facilitating the discovery and the assessment of fitness of use of these resources for a specific mission. There are a number of standards, formats, and APIs that provide metadata, but in order to perform efficient search we need to convert this information into a unified, machine-readable semantic representation that enables the discovery of relevant resources to satisfy the mission of the end user. As an understanding of the kind of metadata information being searched is needed to perform a better

and smarter search, we need a model that accommodates extensions over time without breaking the proposed architecture.

A number of metadata standards have been reviewed, including the W3C standard DCAT, DCAT-AP, GeoDCAT-AP, ADMS, Project Open Data 1.1, Dublin Core, ISO 19115, and ISO 19119 to identify the common and relevant metadata information needed for search and discovery, but also to identify the gaps to address the requirements to describe dataset, service, portrayal information, schema and schema mapping. It quickly emerged that the DCAT standard and its different application profiles were data-centric and insufficient to describe metadata for portrayal information, schemas and services. The goal of this effort was not to define a new standard, but to leverage existing standards to define an application profile of DCAT that could accommodate the schema, service, and portrayal information needed for discovery and filling the gaps by introducing additional properties and fields, all while preserving backward compatibility with existing standards.

This analysis resulted in a new application profile of DCAT called the **Semantic Registry Information Model (SRIM) Core Model**, referred to later in the document as **SRIM Core**. **SRIM Core** is defined as a superset of DCAT and its existing application profiles (DCAT-AP, GeoDCAT-AP, ADMS), and it introduces a superclass of **dcat:Dataset** called **srim:Item**. The ontology draws from multiple well-established standards such as W3C, DCAT, Project Open Data 1.1, DCAT-AP, GeoDCAT-AP, VCard, Dublin Core, and PAV, but also addresses some gaps in the standards, such as descriptions of web services (for example OGC WMS, WFS), richer description for geospatial data, additional metadata to model schema, schema mapping, and portrayal information in order to enable better semantic search of resources that fit the mission of the users. The ontology draws also on the geospatial metadata standard ISO 19115. SRIM enables the integration of different metadata providers (CSW, CKAN, POD WAF, WMS, WCS) by providing a common core vocabulary to describe resources (data, services, vocabularies, map, layers, schemas, etc) and accommodate the specificities of each source by leveraging the built-in extensibility mechanism in OWL. The integration is done through the use of a semantic bridge that maps the syntactic metadata (JSON, XML based) to a semantic representation based on the SRIM model. The SRIM Core model has been extended by introducing the **SRIM application profile** to represent other kinds of geospatial assets such as schemas and portrayal information (see section for Semantic Mediation and Semantic Portrayal Service).

The purpose of SRIM Core is to define a common interchange metadata format for geospatial portals. In order to achieve this, SRIM defines a set of classes and properties, grouped into mandatory, recommended, and optional. Such classes and properties correspond to information on register items and registers that are shared by many data portals, aiding interoperability. Although SRIM is designed to be independent from its actual implementation, RDF [RDF] and Linked Data [LDBOOK] are the reference technologies used to perform the modeling and preserve the semantic fidelity of the conceptual model. However, to facilitate a wide adoption, we provide an encoding based on JSON, which could be converted transparently back to a semantic model using a JSON-LD context. The JSON has been closely aligned with the Project Open Data metadata schema 1.1 standard. However, to accommodate some of the requirements needed by the Semantic Registry Service, we extended, and sometimes modified, the model when needed.



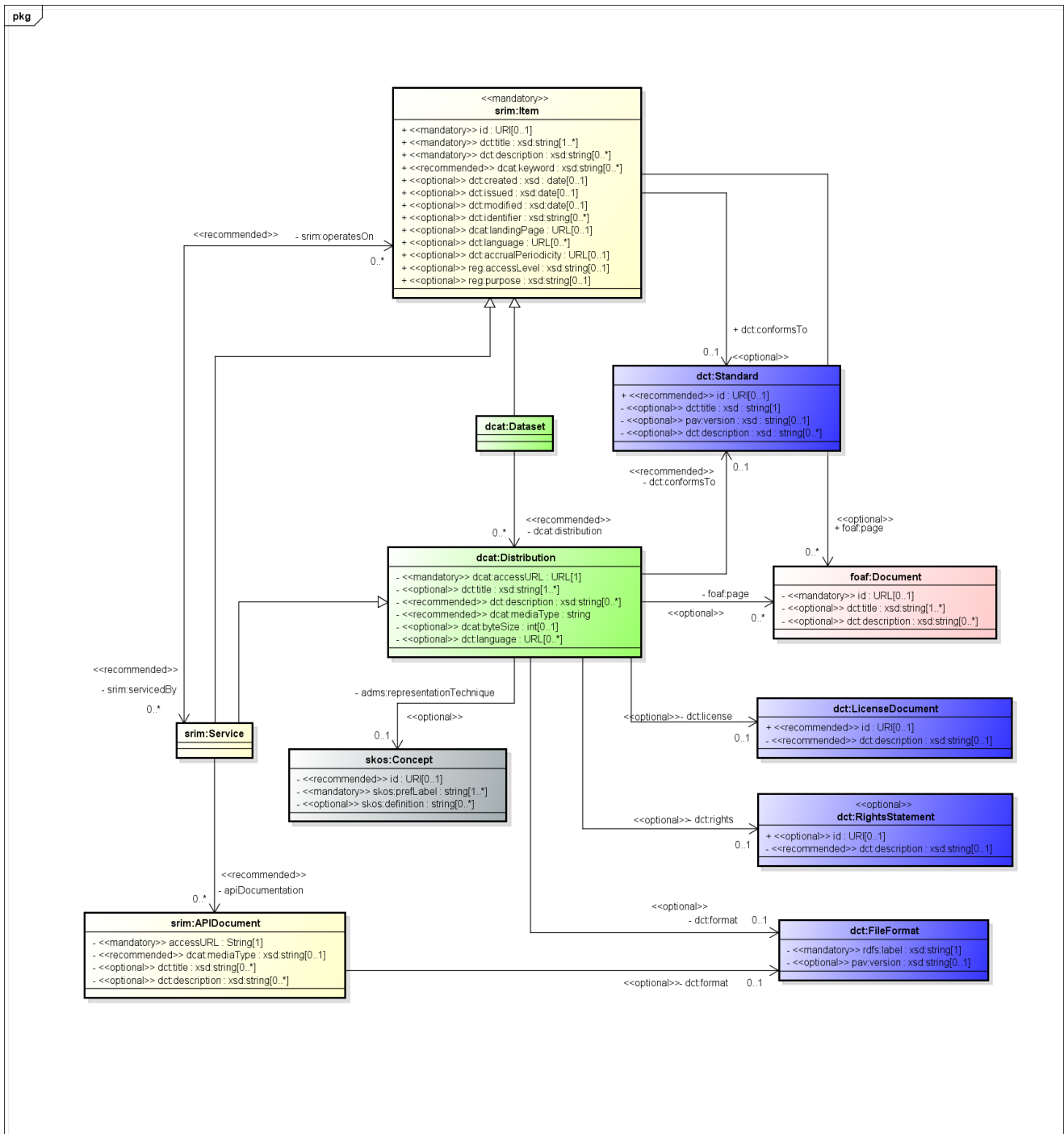


Figure 16. Dataset and Service Application Profile

The following sections describe the details of every class in the model.

## srim:Register

A **Register** is a curated collection of metadata about items stored in a repository (registry) used to enable search and discovery of resources. This class is used as a container to exchange registers of resources between registries. The Register is made accessible through a Registry Service.

**Rationale:** The items are usually organized in catalogs (a synonym of registers). To favor reusability of the registry service for multiple purpose, catalogs can be specialized for a specific register item type (datasets, services, layers, maps, features, vocabularies, etc..) and conform to a specific application profile, or be organized for a specific domain or organization.

## Mandatory properties

Table 3. Register Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
alias	srim:alias	rdfs:Literal	An alias name for the Register that can be used in queries	1
title	dct:title	rdfs:Literal	This property contains a name given to the register. This property can be repeated for parallel language versions of the name.	1..n
description	dct:description	rdfs:Literal	This property contains a free-text account of the Register. This property can be repeated for parallel language versions of the description.	1..n
publisher	dct:publisher	foaf:Agent	This property refers to an entity (organization) responsible for making the register available	1
items	srim:item	srim:Item	This property links the Register with a Resource that is part of the register. This can be modeled as hyperlink in implementation.	1..n
item Class	srim:itemClass	srim:ItemClass	This property links the Register to the item classes it supports.	1..n
application profile	dct:conformsTo	URL	This property refers to an Application Profile that the Register metadata conforms to.	1

## Recommended properties

Table 4. Register Recommended Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the Register (effort should done to make it resolvable)	0..1
landing page	dcat:landingPage	URL	This property refers to a web page that acts as the main page for the Register.	0..1
language	dct:language	URL	This property refers to a language used in the textual metadata describing titles, descriptions, etc. of the resources in the Register. This property can be repeated if the metadata is provided in multiple languages.	0..n
license	dct:license	dct:LicenseDocument	This property refers to the license under which the Register can be used or reused.	0..1

Property	URI	Range	Usage Note	Cardinality
release date	dct:issued	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the date of formal issuance (e.g., publication) of the Register.	0..1
update/modification date	dct:modified	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the most recent date on which the Register was modified.	0..1
theme taxonomies	dcat:themeTaxonomy	skos:ConceptScheme	This property refers to a knowledge organization system (KOS) used to classify the Register's items	0..n

## Optional properties

Table 5. Register Optional Properties

Property	URI	Range	Usage Note	Cardinality
entry	srim:entry	srim:RegisterEntry	This property refers to a <b>RegisterEntry</b> that is part of the Register. This can be modeled as hyperlink in implementation.	0..n
rights	dct:rights	dct:RightsStatement	This property refers to a statement that specifies rights associated with the Register	0..n
spatial/geographic	dct:spatial	dct:Location	This property refers to a geographical area covered by the Register	0..n
temporal	dct:temporal	dct:PeriodOfTime	This property refers to a temporal period covered by the register	0..n

## srim:RegisterEntry

A **RegisterEntry** provides a description of an **Item** entry in the Register. The register entry is used to capture provenance information and the status of the register item within the register. This is mostly a supporting class that is used for the administration of the resources within the registry.

**Rationale:** The Register Entry is a concept needed only for an administrative purpose to manage and understand the lifecycle of the register items. This object is not typically used by end users searching for data (that is why the **RegisterEntry** is made optional in our model). The Register entry corresponds to the **CatalogRecord** in DCAT.

## Mandatory properties

Table 6. RegisterEntry Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	dct:identifier	string	Unique identifier identifying the registry entry within the register	1
primaryTopic	foaf:primaryTopic	<a href="#">srim:Item</a>	This property links the RegisterEntry to the Item described in the entry. This can be modeled as hyperlink in implementation.	1
listing date	dct:issued	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the date on which the description of the resource was included in the Register	1

## Recommended properties

Table 7. RegisterEntry Recommended Properties

Property	URI	Range	Usage Note	Cardinality
update/modification date	dct:modified	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the most recent date on which the RegisterEntry was changed or modified.	0..1
provenance	prov:wasGeneratedBy	<a href="#">prov:Activity</a>	This property refers to provenance information how this registry entry was generated (by harvester, by human editing, by tools).	0..1
status	srim:status	string	This property refers to the type of the latest revision of a resource's entry in the Register. It MUST take one of the values "retired", "superseded", or "valid" depending on whether this latest revision is a result of a creation, update or deprecation. Additional states may be needed for the registry adjudication process such as "proposed" or "unaccepted".	0..1
title	dct:title	rdfs:Literal	This property contains a name given to the RegisterEntry for parallel language versions of the name	0..n

Property	URI	Range	Usage Note	Cardinality
description	dct:description	rdfs:Literal	This property contains a free-text account of the entry. This property can be repeated for parallel language versions of the description.	0..n

### Optional properties

Table 8. RegisterEntry Optional Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the register entry (effort should done to make it resolvable)	0..1
language	dct:language	URL	This property refers to a language used in the textual metadata describing titles, descriptions, etc. of the resources. This property can be repeated if the metadata is provided in multiple languages	0..n
source metadata	dct:source	rdfs:Resource	This property refers to the original metadata that was used in creating metadata for the register item	0..1
previous version	pav:previousVersion	<a href="#">srim:RegisterEntry</a>	Previous version of this entry	0..1

### srim:ItemClass

An ItemClass represents a class of item that can be managed by the registry service. Examples of item classes are datasets, maps, layers, services, vocabularies, styles, and application schemas. Each Item must be associated with one ItemClass. The Registry Service should provide the list of item classes that are supported, and a register can manage one or more item classes.

**Rationale:** In order to validate the type of item submitted to the register, we need to classify the register items according to a well-known domain class. The ItemClass corresponds to an ontology class in the RDF world.

### Mandatory properties

Table 9. ItemClass Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the item class, which should be defined by an OWL ontology. Effort should be made to make the URL resolvable.	1



## Recommended properties

Table 10. ItemClass Recommended Properties

Property	URI	Range	Usage Note	Cardinality
name	srim:name	rdfs:Literal	This property defines a unique name for the item class that could be used in API. This property is required for the registry service implementation	0..1
title	dct:title	rdfs:Literal	This property contains a name given to the Register entry for parallel language versions of the name	0..n
description	dct:description	rdfs:Literal	This property contains a free-text account of the entry. This property can be repeated for parallel language versions of the description.	0..n
Technical standard	srim:technicalStandard	dct:Standard	This property refers to a technical standard the item class conforms to. For example DCAT ontology, DCAT-AP, etc.	0..n

## srim:Item

A register **Item** (also known as register Resource) is the top class of the item hierarchy. It is a subclass of **rdfs:Resource**. It represents any information entities or services that can be registered in a register.

**Rationale:** The **Item** is a generalization of the DCAT Dataset concept. It contains all the common metadata and tradecraft information about any resource of interest for the end-user that needs to be found through a common API.

## Mandatory properties

Table 11. Item Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the resource (effort should done to make it resolvable)	1
type	rdf:type	rdfs:Classes	The RDFS Class that the item instantiates	1
title	dct:title	rdfs:Literal	This property contains a name given to the register item. This property can be repeated for parallel language versions of the name.	1..n
description	dct:description	rdfs:Literal	This property contains a free-text account of the register item. This property can be repeated for parallel language versions of the description.	1..n

## Recommended properties

Table 12. Item Recommended Properties

Property	URI	Range	Usage Note	Cardinality
contactPoint	dcat:contactPoint	<a href="#">vcard:VCard</a>	This property contains contact information that can be used for sending comments about the register item	0..n
publisher	dct:publisher	<a href="#">foaf:Agent</a>	This property refers to an entity (organization) responsible for making the register item available	0..n
keyword/tag	dcat:keyword	rdfs:Literal	This property contains a keyword or tag describing the register item.	0..n
theme/category	dcat:theme, subproperty of dct:subject	<a href="#">skos:Concept</a>	This property refers to a category of the register item. A register item may be associated with multiple themes.	0..n
creation date	dct:created	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the date of creation of the register item.	0..1
release date	dct:issued	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the date of formal issuance (e.g., publication) of the register item.	0..1
update/modification date	dct:modified	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the most recent date on which the register item was modified.	0..1

## Optional properties

Table 13. Item Optional Properties

Property	URI	Range	Usage Note	Cardinality
Third party identifier	dct:identifier	xsd:string	This property contains the main identifier for the register item, e.g. the URI or other unique identifier in the context of the Register.	0..n
other identifier	id:hasIdentifier	id:Identifier	This property refers to a secondary identifier of the register item, such as MAST/ADS18 , DataCite19 , DOI20 , EZID21 or W3ID22 .	0..n
version	pav:hasVersion	srim:Release	This property refers to a version of the register item, and stores a version number and release notes in addition to other version information.	0..1
current version	pav:hasCurrentVersion	srim:Release	This property refers to the current version of the register item, and stores a version number and release notes in addition to other version information. The current version of the item must also be stored under the previous version property.	0..n
type	dct:type (resource Type in JSON)	skos:Concept	This property refers to the type of the register item defined in a controlled vocabulary.	0..n
spatial/geographic	dct:spatial	dct:Location	This property refers to a geographical area covered by the register item.	0..n
temporal	dct:temporal	dct:PeriodOfTime	This property refers to a temporal period covered by the register item	0..n
landingPage	dcat:landingPage	foaf:Document	This property refers to a web page that provides access to the register item, its Distributions and/or additional information. It is intended to point to a landing page at the original register item provider, not to a page on a site of a third party, such as an aggregator	0..n
language	dct:language	URL	This property refers to a language of the register item. This property can be repeated if there are multiple languages in the register item.	0..n
owner	dct:rightsHolder	foaf:Agent	This property refers to the owner of the source	0..n
creator	dct:creator	foaf:Agent	This property refers to the creator of the source	0..n
contributor	dct:contributor	foaf:Agent	This property refers to a contributor to the source	0..n

Property	URI	Range	Usage Note	Cardinality
other attributions	prov:qualifiedAttribution (attribution in JSON)	prov:Attribution	This property refers to other responsible parties of the register item playing a role that is different than contactPoint, creator, rightsHolder or publisher.	0..n
purpose	srим:purpose	skos:Concept	The purpose of the item	0..n
audience	srим:audience	skos:Concept	Intended audience for the item	0..n]
topic	srим:topic	skos:Concept	The central topic the item describes	0..n
project	srим:project	foaf:Project	The project this item is produced for	0..n
function	srим:function	skos:Concept	The function the item is intended for	0..n
link	link:hasLink	link:Link	This property refers to a resource related to the described register item that is a link to a web page, file, or other such resource on the web. Used when the relationship between the Item and the content of the link is important.	0..n
Relation	dct:relation	rdfs:Resource	This property is used when this register item has a relation to another resource that is web accessible. If the role of the resource is important, use link:hasLink, or use subproperty of dct:relation	0..n
access level	srим:accessLevel	string	Access Level of the Map, Must be one of the following: “public”, “restricted public”, “non-public”	0..1
frequency	dct:accrualPeriodicity	URL	This property refers to the frequency at which the register item is updated. Use the standard vocabulary from <a href="http://purl.org/cld/freq">http://purl.org/cld/freq</a> (example <a href="http://purl.org/cld/freq/irregular">http://purl.org/cld/freq/irregular</a> )	0..1
Item application profile	dct:conformsTo	URL	This property refers to an Application Profile that the Register item conforms to.	0..1
geographic extent	extent:hasGeographicExtent	extent:GeographicExtent	This property refers to the geographic extent of the metadata held inside the Item.	0..n
depiction	foaf:depiction	foaf:Document	This property refers to an image representing the content of the Item.	0..n

Property	URI	Range	Usage Note	Cardinality
provenance statement	dct:provenance	dct:ProvenanceStatement	This property refers a provenance statement about the contents of the Item.	0..n

## prov:Activity

An **Activity** holds the URI of the source that generated a register entry.

**Rationale:** Provides provenance information about where the entry came from.

### Optional Properties

Table 14. Activity Optional Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the generating activity (effort should be done to make it resolvable)	0..1

## vcard:Address

To represent address information, we use the well-known ontology VCard.

**Rationale:** We need a way to describe address information that is based on a well-known standard, so person/organization in charge of resources can be contacted or located.

### Recommended Properties

Table 15. Address Recommended Properties

Property	URI	Range	Usage Note	Cardinality
street-address	vcard:street-address	string	The street address component of the address	0..1
locality	vcard:locality	string	The locality (city, village) component of the address	0..1
region	vcard:region	string	The region (province/state) component of the address	0..1
postal code	vcard:postal-code	string	The postal code component of the address	0..1
country name	vcard:country-name	string	The country name component of the address	0..1

## foaf:Agent

An entity that is associated with Registers and/or Resources. If the Agent is an organization, the use of the Organization ontology is recommended.

**Rationale:** Defining a standard way to represent agents helps to create hub of information around organizations and people.

### Mandatory properties

Table 16. Agent Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
name	foaf:name	rdfs:Literal	This property contains a name of the agent. This property can repeated for different versions of the name (e.g. the name in different languages)	1..n

### Recommended properties

Table 17. Agent Recommended Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the agent (effort should done to make it resolvable)	0..1

## prov:Attribution

Registered resources are attributed to one or more responsibility parties playing different **Attribution** roles (contact point,publisher, rights holder, processor...).

**Rationale:** Needed to understand the role of each contributor of a resources. Provide an extensible mechanism to accommodate new roles.

### Mandatory properties

Table 18. Attribution Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
agent	prov:agent	foaf:Agent	This property defines the agent of the attribution.	1
role	prov:role	URL	The role of the agent in the attribution. Role are typically defined in a taxonomy (such as the one defined in ISO 19115).	1

## skos:Concept

A **Concept** can be viewed as an idea or notion; a unit of thought. However, what constitutes a unit of thought is subjective, and this definition is meant to be suggestive, rather than restrictive.

**Rationale:** The use of controlled vocabularies helps to organize information at the conceptual level and enable semantic inferencing and classification of information that helps the end-user to find the relevant information that fits better his needs.

### Mandatory Properties

Table 19. Concept Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the concept (effort should done to make it resolvable)	1
preferred label	skos:prefLabel	rdfs:Literal	This property contains a preferred label of the concept. This property can be repeated for parallel language versions of the label.	1..n

### Recommended Properties

Table 20. Concept Recommended Properties

Property	URI	Range	Usage Note	Cardinality
type	rdf:type	rdfs:Class or owl:Class subclass of <a href="#">skos:Concept</a>	By default the class is skos:Concept.	0..1
scheme	skos:inScheme	<a href="#">skos:ConceptScheme</a>	This property contains the concept scheme in which this concept is defined.	0..1

### Optional Properties

Table 21. Concept Optional Properties

Property	URI	Range	Usage Note	Cardinality
description	skos:definition	rdfs:Literal	This property contains the definition of the concept. This property can be repeated for parallel language versions of the definition.	0..n

## skos:ConceptScheme

A concept scheme (e.g. controlled vocabulary) in which the Concept is defined. A concept scheme can be viewed as an aggregation of one or more SKOS concepts. Semantic relationships (links) between those concepts may also be viewed as part of a concept scheme.

**Rationale:** The use of controlled vocabularies helps to organize information at the conceptual level and enable semantic inferencing and classification of information that helps the end-user to find the relevant information that fits better his needs.

### Mandatory Properties

Table 22. ConceptScheme Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the concept scheme (effort should done to make it resolvable)	1
title	dct:title	rdfs:Literal	This property contains a title of the concept scheme. This property can be repeated for parallel language versions of the label.	1..n

### Recommended Properties

Table 23. ConceptScheme Recommended Properties

Property	URI	Range	Usage Note	Cardinality
description	skos:definition	rdfs:Literal	This property contains the definition of the concept scheme. This property can be repeated for parallel language versions of the definition.	0..n

## foaf:Document

The **Document** class represents those things which are, broadly conceived, 'documents'. The foaf:Image class is a sub-class of Document, since all images are documents. Document needs to be identified by a resolvable URL on the web.

**Rationale:** We need a way to describe documents to understand their role in the ecosystem of items managed in the registry.

### Mandatory properties

Table 24. Document Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URL	The URL of a well-known Document that is resolvable	1



## Optional properties

Table 25. Document Optional Properties

Property	URI	Range	Usage Note	Cardinality
title	dct:title	rdfs:Literal	This property contains a title for the document. This property can be repeated for parallel language versions of the title.	0..n
description	dct:description	rdfs:Literal	This property contains a free-text account of the document. This property can be repeated for parallel language versions of the description.	0..n

## vcard:Email

To represent email information, we use the well-known ontology VCard.

**Rationale:** We need a way to describe email information that is based on a well-known standard, so the person/organization in charge of a resource can be contacted or located.

## Mandatory properties

Table 26. Email Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
email address	vcard:hasValue	URI	This property holds the email encoded as a URI using the prefix "mailto:" (example mailto:johndoe@example.com)	1

## extent:GeographicExtent

GeographicExtent describes the spatial extent of domain of application of an item and is standardized in WGS 84 Lat/Long coordinate system.

**Rationale:** There is no consensus and common vocabulary to describe spatial in the community. GML Envelope was proposed by it is too cumbersome to process. We introduce four separate fields for each bound (west, east, north and south) that removes any ambiguity and make it easy to query in any spatial index.

## Mandatory properties

Table 27. GeographicExtent Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
northBoundLatitude	extent:northBoundLatitude	xsd:decimal	North bound latitude in decimal degrees	1
southBoundLatitude	extent:southBoundLatitude	xsd:decimal	South bound latitude in decimal degrees	1
westBoundLongitude	extent:westBoundLongitude	xsd:decimal	West bound longitude in decimal degrees	1
eastBoundLongitude	extent:eastBoundLongitude	xsd:decimal	East bound longitude in decimal degrees	1

## id:Identifier

An **Identifier** provides a mechanism to uniquely identify a resource using a given codespace and defined by an authority.

**Rationale:** The support of identifiers helps to find items based on some well-know identification systems (ISBN for example).

### Mandatory Properties

*Table 28. Identifier Mandatory Properties*

Property	URI	Range	Usage Note	Cardinality
code	id:code	string	The unique code associated with the identifier	1
codespace	id:codespace	string	The codespace in which the code is defined (ISBN for example)	1

### Recommended Properties

*Table 29. Identifier Recommended Properties*

Property	URI	Range	Usage Note	Cardinality
authority	id:authority	URL	The authority defining the identifier scheme	0..1

## dct:LicenseDocument

### Recommended Properties

Table 30. LicenseDocument Recommended Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the license document (effort should be done to make it resolvable)	0..1
description	dct:description	string	This property stores the content of the license document as a string	0..1

## link:Link

A **Link** contains information about a resource that is related to an Item or another entity and that can be found on the Internet.

**Rationale:** A Link is useful as a container for a rdfs:Resource that can store the relationship of a link to its parent item as well as information about what is on the other side of the URL.

### Mandatory properties

Table 31. Link Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
link to resource	link:href	rdfs:Resource	This property stores a link to whatever content is being referenced.	1
relationship to Item	link:rel	skos:Concept	This property refers to a Concept representing the relationship between the linked resource and the Link's parent Item	1

### Recommended properties

Table 32. Link Recommended Properties

Property	URI	Range	Usage Note	Cardinality
title	dct:title	string	This property refers to the name given to a link.	0..1
description	dct:description	rdfs:Literal	Refers to the description given to a link. This property can be repeated for parallel language versions of the description.	0..n
format	dct:format	dct:MediaType	This property refers to the format of the link's content.	0..1

## Optional properties

Table 33. Link Optional Properties

Property	URI	Range	Usage Note	Cardinality
media type	link:mediaType	string	MIME media type of the link's content.	0..1
language	link:hrefLang	string	Language of a link's content. This property can be repeated if there is more than one language.	0..n

## dct:Location

**Location** represents a place that is related to the item stored in the register. Each location should have a geographic extent expressed in WGS84 to enable spatial search in a consistent way. We use **dct:Location** as the base class for Location.

**Rationale:** Essential for spatial scoping of data. Location should be referred to as much as possible with a resolvable URI that points to a controlled vocabulary of place names (Gazetteer). This allows spatial reasoning using place part-of relationships. We try to keep the definition of Location to a bare minimum. More complex geometry can be added down the road. We may use W3c Location ontology to extend the model.

## Recommended properties

Table 34. Location Recommended Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the location (effort should be done to make it resolvable) defined in a gazetteer such as Geonames or DBpedia	0..1
location name	locn:geographicName	rdfs:Literal	A geographic name is a proper noun applied to a spatial object. This property can be repeated for parallel language versions of the place name.	0..n
description	dct:description	string	This property contains a free-text account of the location. This property can be repeated for parallel language versions of the description.	0..n

## gr:OpeningHoursSpecification

An **OpeningHoursSpecification** describes the hours that a business or other organization is open during the week. It is based in the well-known GoodRelations vocabulary.

**Rationale:** Having a standard way to capture the opening hours of a business or organization helps understand when it is best to visit or otherwise contact said business or organization.

## Optional properties

Table 35. *OpeningHoursSpecification* Optional Properties

Property	URI	Range	Usage Note	Cardinality
label	<code>rdfs:label</code>	string	This property stores opening hours information when it cannot be converted to a standardized format.	0..1
opens	<code>gr:opens</code>	<code>xsd:time</code>	This property stores the time at which the organization opens.	0..1
closes	<code>gr:closes</code>	<code>xsd:time</code>	This property stores the time at which the organization closes.	0..1
opening days of week	<code>gr:hasOpeningHoursDayOfWeek</code>	<code>gr:DayOfWeek</code>	This property stores the days of the week on which the organization is open. These are stored as elements of the controlled <code>gr:DayOfWeek</code> vocabulary.	0..n

## org:Organizaton

An **Organization** represents a collection of people organized together into a community or other social, commercial or political structure. The group has some common purpose or reason for existence which goes beyond the set of people belonging to it and can act as an Agent. Organizations are often decomposable into hierarchical structures. An Organization is a specialization of an Agent. It is strongly advised that the Organization provides a URL that is shareable and stable according to the best practices of Linked Data.

**Rationale:** Defining a standard way to represent organizations helps to create hub of information around organizations.

## Recommended properties

Table 36. *Organization* Recommended Properties

Property	URI	Range	Usage Note	Cardinality
parent organization	<code>org:subOrganizationOf</code>	<a href="#">org:Organization</a>	The parent organization of the organization	0..1

## dct:PeriodOfTime

A **PeriodOfTime** denotes the temporal scope of a register item. We use `dct:PeriodOfTime` as the base class, and a time period has `startDate` and/or `endDate`. If they are the same, they represent a temporal instant (point in time). A `PeriodOfTime` can also refer to a URI that can be resolvable and is defined by a well-known taxonomy of periods of time.

**Rationale:** Essential for temporal scoping of data.

## Recommended properties

Table 37. *PeriodOfTime Recommended Properties*

Property	URI	Range	Usage Note	Cardinality
Start Date	schema:startDate	xsd:date	The start date of the time period	0..1
End Date	schema:endDate	xsd:date	The end date of the time period	0..1

## Optional properties

Table 38. *PeriodOfTime Optional Properties*

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the location (effort should done to make it resolvable) defined in a controlled vocabulary such as Getty	0..1

## vcard:Phone

To represent phone information, we use the well-known ontology VCard.

**Rationale:** We need a way to describe phone information that is based on a well-known standard, so the person/organization in charge of a resource can be contacted or located.

## Mandatory properties

Table 39. *Phone Mandatory Properties*

Property	URI	Range	Usage Note	Cardinality
phone number	vcard:hasValue	URI	This property holds the telephone number encoded as a URI using the prefix "tel:" (example tel:+1 999-999-999)	1

## foaf:Project

A project for which data was generated or collected. Uses the well-known **foaf:Project** ontology.

**Rationale:** Provides a way to store the project for which the metadata in the register item was generated.

## Recommended Properties

Table 40. *Project Recommended Properties*

Property	URI	Range	Usage Note	Cardinality
id	rdf:id	URI	The URI of the generating project (effort should be done to make it resolvable)	0..1
name	foaf:name	rdfs:Literal	This property contains a title of the project. This property can be repeated for parallel language versions of the title.	0..n

## dct:ProvenanceStatement

Statement containing a provenance statement about the origin of a register item. Based on the well-known DCTerms provenance statement (dct:ProvenanceStatement).

**Rationale:** Provides a way to store free-form provenance information.

### Mandatory Properties

Table 41. ProvenanceStatement Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
description	dct:description	rdfs:Literal	This property contains a provenance statement in free-text form. This property can be repeated for parallel language versions of the description.	0..n

### Optional Properties

Table 42. ProvenanceStatement Optional Properties

Property	URI	Range	Usage Note	Cardinality
title	dct:title	string	This property stores the title of the provenance statement.	0..1
identifier	rdf:id	URI	The URI of the provenance statement (effort should be done to make it resolvable)	0..1

## srim:Release

A **Release** contains information detailing a specific version of an item.

**Rationale:** In order to keep track of versioning in sufficient detail, we need to have a class that stores version information such as version number and release notes.

### Mandatory Properties

Table 43. Release Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the resource (effort should done to make it resolvable)	1
version	pav:version	string	This property refers to the version number of the release.	1

## Recommended Properties

Table 44. Release Recommended Properties

Property	URI	Range	Usage Note	Cardinality
version notes	adms:versionNotes	rdfs:Literal	Version notes pertaining to the release. This property can be repeated for parallel language versions of the notes.	0..n
release date	dct:issued	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the date of formal issuance (e.g., publication) of the release.	0..1

## Optional Properties

Table 45. Release Optional Properties

Property	URI	Range	Usage Note	Cardinality
parent item	pav:versionOf	URI	Unique identifier of the item to which this release pertains.	0..1

## dct:RightsStatement

A statement about the intellectual property rights (IPR) held in or over a resource, a legal document giving official permission to do something with a resource, or a statement about access rights.

**Rationale:** Important legal information protecting intellectual properties.

## Recommended Properties

Table 46. RightsStatement Recommended Properties



Property	URI	Range	Usage Note	Cardinality
description	skos:definition	rdfs:Literal	This property contains the definition of the rights statement. This property can be repeated for parallel language versions of the definition.	0..n

### Optional Properties

Table 47. RightsStatement Optional Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of a well-known rights statement that is resolvable.	0..1

### dct:Standard

A **Standard** provides a basis for comparison; a reference point against which other things can be evaluated.

**Rationale:** This is an important concept that is used to understand how register items conforms to standards.

### Recommended properties

Table 48. Standard Recommended Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the standard (effort should done to make it resolvable)	0..1

### Optional properties

Table 49. Standard Optional Properties

Property	URI	Range	Usage Note	Cardinality
title	dct:title	string	This property refers to the title of the standard	0..1
version	pav:version	string	This property refers to a specific version of this standard	0..1
description	dct:description	rdfs:Literal	This property contains a description of the standard. This property can be repeated for parallel language versions of the standard.	0..n

## vcard:VCard

To represent contact information, we use the well-known ontology **VCard**.

**Rationale:** We need a way to describe contact information that is based on a well-known standard, so person/organization in charge of resources can be contacted to perform corrections or status information when necessary.

### Mandatory properties

Table 50. VCard Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
full name	vcard:fn	string	The name of the person who is the main contact point	1
email	vcard:hasEmail	URL	The email address encoded as a URL starting with the prefix "mailto:" (mailto:foo@example.com). It is strongly recommended to set this property to have a way to contact the person in charge of a register item	1..n

### Optional properties

Table 51. VCard Optional Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the vcard	0..1
organization-name	vcard:organization-name	String	The name of the organization of the contact	0..1
telephone	vcard:hasTelephone	<a href="#">vcard:Phone</a>	This property contains a Phone object that holds the telephone number encoded as a URL using the prefix tel: (example tel:+1 999-999-999)	0..n
title	vcard:title	string	The title (position) of the contact	0..1
landing page	dcat:landingPage	URL	A URL pointing to the landing page of the contact point.	0..1
address	vcard:address	<a href="#">vcard:Address</a>	The address associated with the Vcard	0..1
opening hours	gr:hasOpeningHoursSpecification	<a href="#">gr:OpeningHoursSpecification</a>	The opening hours of the organization described in the vcard.	0..1

# Dataset Application Profile

## dcat:Dataset

A **Dataset** is a collection of data, published or curated by a single agent and available for access or download in one or more formats.

**Rationale:** A register item may be a dataset, in which case it is necessary to store information about the dataset and the data inside it.

### Mandatory properties

Table 52. Dataset Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
distribution	dcat:distribution	dcat:Distribution	This property refers to distributions relevant to the dataset	1..n

## dcat:Distribution

A **Distribution** is piece of data or metadata than can be accessed via a URL and is relevant to a [Dataset](#).

**Rationale:** If a register item is a dataset, it is necessary to store information about the data inside it.

### Mandatory properties

Table 53. Distribution Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
access URL	dcat:accessURL	URL	This property stores a URL pointing to the content of the distribution	1

### Recommended properties

Table 54. Distribution Recommended Properties

Property	URI	Range	Usage Note	Cardinality
media type	dcat:mediaType	string	This property holds the media type of the contents of the distribution.	0..1
description	dct:description	string	This property stores a description of the distribution. Cardinality is unbounded to allow for multiple languages.	0..n
standard	dct:conformsTo	dct:Standard	The standard a distribution conforms to.	0..1

## Optional properties

Table 55. Distribution Optional Properties

Property	URI	Range	Usage Note	Cardinality
title	dct:title	string	The title of the distribution. Cardinality is unbounded to allow for multiple languages.	0..n
byte size	dcat:byteSize	integer	The size of the distribution file in bytes, if applicable.	0..1
language	dct:language	URL	This property refers to a language used in the distribution describing titles, descriptions, etc. This property can be repeated if the distribution is provided in multiple languages.	0..n
format	dct:format	<a href="#">dct:FileFormat</a>	The file format of the distribution, if applicable.	0..1
document	foaf:page	<a href="#">foaf:Document</a>	A document relevant to the distribution.	0..n
license document	dct:license	<a href="#">dct:LicenseDocument</a>	A license document pertaining to the distribution, specifying under what circumstances it can be used or reused.	0..n
rights statement	dct:rights	<a href="#">dct:RightsStatement</a>	A rights statement pertaining to the distribution.	0..n
representation technique	adms:representationTechnique	<a href="#">skos:Concept</a>	This property refers to the technique used to represent the distribution.	0..n

## dct:FileFormat

A **FileFormat** stores information about a file format and is defined by the well-known DCTerms (Dublin Core) vocabulary.

**Rationale:** Distributions are often stored as a downloadable file, and it is important to keep track of what format each file is in.

## Mandatory properties

Table 56. FileFormat Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
format name	rdfs:label	string	This property stores the name of the file format	1

## Optional properties

Table 57. FileFormat Optional Properties

Property	URI	Range	Usage Note	Cardinality
format version	pav:version	string	The version of the format that is being used	0..1

# Service Application Profile

## srim:Service

A Web **Service** is a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web. In a web service, web technology such as the HTTP, originally designed for human-to-machine communication, is utilized for machine-to-machine communication, more specifically for transferring machine readable file formats such as XML and JSON. Service has all the properties of both a [Distribution](#) and an [Item](#).

**Rationale:** An important aspect of the registry model is to support a service description that can be accessed automatically by machines. We want a core minimal set of properties that enables this automation.

## Recommended properties

Table 58. Service Recommended Properties

Property	URI	Range	Usage Note	Cardinality
relevant Item	srim:operatesOn	<a href="#">srim:Item</a>	This property contains the Item that the service operates on.	0..n

## srim:APIDocument

An **APIDocument** describes a web accessible document that describes the API of the service.

**Rationale:** To be able to automate the access to a service that is not adhering to a well-known standard (such OGC WMS, ArcREST), we need a mechanism to point out to human and machine processable API document such as (RAML, Swagger, HAL Profile, ALPS profile).

## Mandatory properties

Table 59. APIDocument Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
Access URL	dcat:accessURL	URL	This property contains the URL to access the API document that can be processed by client	1

## Recommended properties

Table 60. APIDocument Recommended Properties

Property	URI	Range	Usage Note	Cardinality
Media type	dcat:mediaType	Mime type string	This property contains the media type for the API document	0..1

## Optional properties

Table 61. APIDocument Optional Properties

Property	URI	Range	Usage Note	Cardinality
format	dcat:format	<a href="#">dct:FileFormat</a>	This property contains a human description of the format	0..1
title	dct:title	rdfs:Literal	This property contains a title of the API documentation. This property can be repeated for parallel language versions of the label.	0..n
description	dct:description	rdfs:Literal	This property contains a free-text account of the API documentation. This property can be repeated for parallel language versions of the description.	0..n

# Appendix B: SRIM Schema Application Profile

In the following sections, classes and properties are grouped under headings ‘mandatory’, ‘recommended’ and ‘optional’. These terms have the following meaning.

- **Mandatory class:** a receiver of data **MUST** be able to process information about instances of the class; a sender of data **MUST** provide information about instances of the class.
- **Recommended class:** a sender of data **SHOULD** provide information about instances of the class; a sender of data **MUST** provide information about instances of the class, if such information is available; a receiver of data **MUST** be able to process information about instances of the class.
- **Optional class:** a receiver **MUST** be able to process information about instances of the class; a sender **MAY** provide the information but is not obliged to do so.
- **Mandatory property:** a receiver **MUST** be able to process the information for that property; a sender **MUST** provide the information for that property.
- **Recommended property:** a receiver **MUST** be able to process the information for that property; a sender **SHOULD** provide the information for that property if it is available.
- **Optional property:** a receiver **MUST** be able to process the information for that property; a sender **MAY** provide the information for that property but is not obliged to do so.

The meaning of the terms **MUST**, **MUST NOT**, **SHOULD** and **MAY** in this section and in the following sections are as defined in RFC 2119.

In the given context, the term "processing" means that receivers must accept incoming data and transparently provide these data to applications and services. It does neither imply nor prescribe what applications and services finally do with the data (parse, convert, store, make searchable, display to users, etc.).

Classes are classified as **Mandatory** if they appear as the range of one of the mandatory properties.

## Semantic Mediation Ontology

### Overview

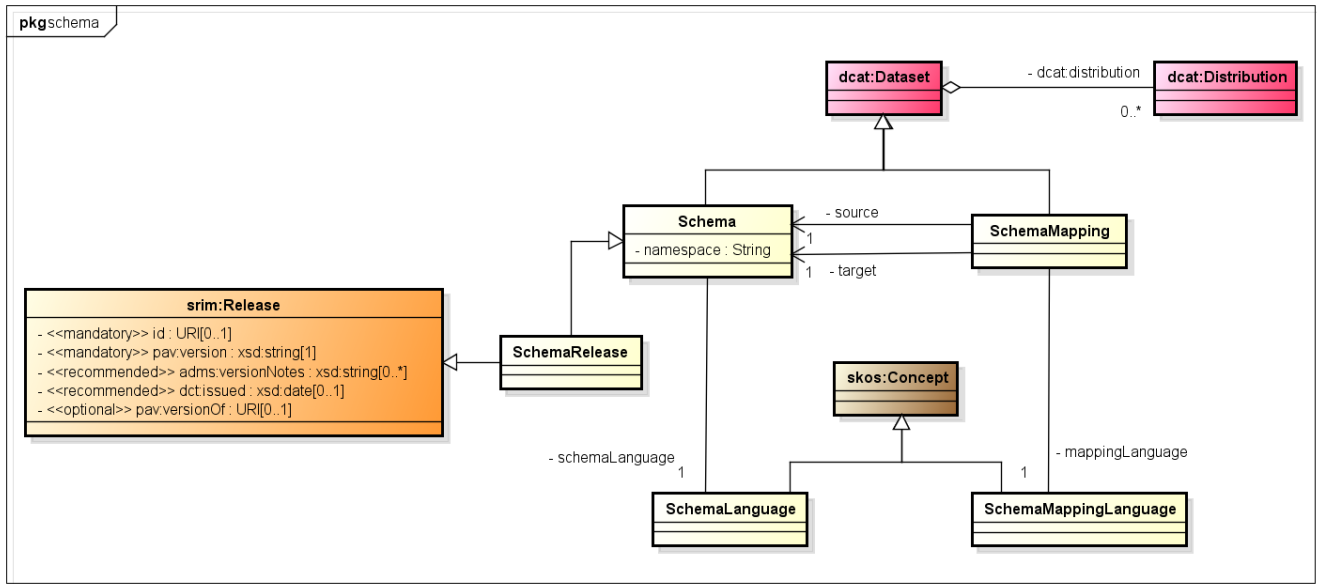


Figure 17. Core Semantic Mediation Model

## schema:Schema

A **Schema** in this context is a specific format used to represent information. Schema extends [dcat:Dataset](#).

**Rationale:** Because the same data can be represented in a number of different ways, we want to be able to determine what schema is being used and convert from one schema to another.

### Mandatory properties

Table 62. Schema Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the resource (effort should done to make it resolvable)	1
type	rdf:type	rdfs:Classes	The RDFS Class that the item instantiates (schema:Schema or future subclasses)	1
title	dct:title	rdfs:Literal	This property contains a title given to the schema. This property can be repeated for parallel language versions of the name.	1..n
description	dct:description	rdfs:Literal	This property contains a free-text account of the schema. This property can be repeated for parallel language versions of the description.	1..n

### Recommended properties

Table 63. Schema Recommended Properties



Property	URI	Range	Usage Note	Cardinality
namespace	schema:namespace	string	This property store the namespace of the schema when available	0..1
schema language	schema:schemaLanguage	<a href="#">schema:SchemaLanguage</a>	This property holds the language used in the schema if applicable.	1
distribution	dcat:distribution	<a href="#">dcat:Distribution</a>	This property stores distributions held within or otherwise relevant to the schema	1..n

### Example

```

@prefix org: <http://www.socialml.org/ontologies/organization#> .
@prefix link: <http://www.opengis.net/ont/link#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema-org: <http://schema.org/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix srin: <http://www.opengis.net/ont/testbed12/srin#> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

```

```

<https://www.fgdc.gov/schemas/metadata/fgdc-std-001-1998>
  a schema:Schema ;
  dct:created "2016-10-10T16:10:01.213Z"^^xsd:dateTime ;
  dct:description "The objectives of the standard are to provide a common
set of terminology and definitions for the documentation of digital geospatial data.
The standard establishes the names of data elements and compound elements (groups of
data elements) to be used for these purposes, the definitions of these compound
elements and data elements, and information about the values that are to be provided
for the data elements." ;
  dct:identifier "FGDC-STD-001-1998" ;
  dct:publisher <http://www.fgdc.gov> ;
  dct:title "Content Standards for Digital Geospatial Metadata
Version 2.0" ;
  schema:schemaLanguage <http://www.w3.org/TR/xmlschema-1/> ;
  dcat:distribution [ a dcat:Distribution ;
dct:description "This is the XML Schema
for formal metadata, metadata conforming to the Content Standards for Digital
Geospatial Metadata of the Federal Geographic Data Committee. This schema corresponds
to the June,\t 1998 version of the standard, FGDC-STD-001-1998.\t This file is the
primary XML Schema and loads the definitions for sections 1-10 of the standard from
separate schema modules." ;
dct:title "FGDC CSDGM 2.0
Metadata XML Schema" ;
adms:representationTechnique
<http://www.w3.org/TR/xmlschema-1/> ;
dcat:accessURL
<https://www.fgdc.gov/schemas/metadata/fgdc-std-001-1998.xsd>
] ;
  dcat:keyword "FGDC" , "XML Schema" , "CSGDM" , "Geospatial" .

<http://www.fgdc.gov>
  a org:Organization ;
  foaf:name "Federal Geographic Data Committee (FGDC)" .

```

## schema:SchemaMapping

A **SchemaMapping** is a mapping used to convert a piece of data from one schema to another. SchemaMapping extends [dcat:Dataset](#).

**Rationale:** Because there is a need to transform data from one scheme to another, it is useful to be able to keep track of the schemas being mapped between.

### Mandatory properties

Table 64. SchemaMapping Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the resource (effort should done to make it resolvable)	1
type	rdf:type	rdfs:Classes	The RDFS Class that the item instantiates (schema:SchemaMapping or future subclasses)	1
title	dct:title	rdfs:Literal	This property contains a title given to the schema. This property can be repeated for parallel language versions of the name.	1..n
description	dct:description	rdfs:Literal	This property contains a free-text account of the schema. This property can be repeated for parallel language versions of the description.	1..n
source schema	schema:source	<a href="#">schema:Schema</a>	This property stores the schema being mapped <i>from</i> .	1
target schema	schema:target	<a href="#">schema:Schema</a>	This property holds the schema being mapped <i>to</i> .	1
mapping language	schema:mappingLanguage	<a href="#">schema:SchemaMappingLanguage</a>	This property stores the language used to create a schema mapping.	1

### Example

```

@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .
@prefix extent: <http://www.opengis.net/ont/extent#> .
@prefix adms: <http://www.w3.org/TR/vocab-adms/#> .
@prefix pav: <http://purl.org/pav/> .
@prefix org: <http://www.socialml.org/ontologies/organization#> .
@prefix link: <http://www.opengis.net/ont/link#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema-org: <http://schema.org/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix srim: <http://www.opengis.net/ont/testbed12/srim#> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

```

```

<http://www.opengis.net/testbed12/schemas#MNIS2DDMS>
  a schema:SchemaMapping ;
  dct:created "2016-10-10T16:10:06.490Z"^^xsd:dateTime ;
  dct:description "Schema Mapping from NMIS 2.2 to DDMS" ;
  dct:publisher <http://www.nga.mil> ;
  dct:title "NMIS 2.2 to DDMS SchemaMapping" ;
  schema:mappingLanguage <https://www.w3.org/TR/xslt20/> ;
  schema:sourceSchema
<http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2> ;
  schema:targetSchema <http://metadata.dod.mil/mdr/ns/DDMS/2.0> ;
  dcat:distribution [ a dcat:Distribution ;
                    dct:description "NGA.STND.0018_2.2 to
DDMS SchemaMapping XSLT Distribution" ;
                    dct:title "NMIS to DDMS
SchemaMapping XSLT Distribution" ;
                    adms:representationTechnique
<https://www.w3.org/TR/xslt20/> ;
                    dcat:accessURL
<http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&service=CSW-
ebRIM&id=urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4>
                    ] ;
  dcat:keyword "NAS" , "DDMS" , "XSLT" , "Geospatial" , "NMIS" .

<http://www.nga.mil>
  a org:Organization ;
  foaf:name "National Geospatial Intelligence Agency (NGA)" .

```

## schema:SchemaLanguage

A **SchemaLanguage** is the language that a schema is written in. SchemaLanguage extends

[skos:Concept](#).

**Rationale:** It is important to keep track of what language a schema is written in so that the dataset using that schema can later be processed.

### Mandatory properties

Table 65. *SchemaLanguage* Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the resource (effort should done to make it resolvable)	1

## schema:SchemaMappingLanguage

A **SchemaMappingLanguage** is the language that defines a schema mapping. *SchemaMappingLanguage* extends [skos:Concept](#).

**Rationale:** It is important to keep track of what language is being used to transform a dataset from one schema to another.

### Mandatory properties

Table 66. *SchemaMappingLanguage* Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the resource (effort should done to make it resolvable)	1

## schema:SchemaRelease

A **SchemaRelease** is a single release (version) of a schema. *SchemaRelease* extends [srim:Release](#).

**Rationale:** It is necessary to keep track of past releases of a schema in case backwards compatability is needed.

### Mandatory Properties

Table 67. *SchemaRelease* Mandatory Properties

Property	URI	Range	Usage Note	Cardinality
identifier	rdf:id	URI	The URI of the resource (effort should done to make it resolvable)	1
version	pav:version	string	This property refers to the version number of the release.	1

## Recommended Properties

Table 68. SchemaRelease Recommended Properties

Property	URI	Range	Usage Note	Cardinality
version notes	adms:versionNotes	rdfs:Literal	Version notes pertaining to the release. This property can be repeated for parallel language versions of the notes.	0..n
release date	dct:issued	rdfs:Literal typed as xsd:date or xsd:dateTime	This property contains the date of formal issuance (e.g., publication) of the release.	0..1

# Appendix C: Semantic Portrayal Ontologies

## A.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, Styled Layer Descriptor Profile of WMS, SVG, KML, CartoCSS, MapCSS).

To favor reusability, the Portrayal ontologies are decomposed into four microtheories, each one addressing an orthogonal aspect of portrayal:

- **Style ontology:** defines the concept of Style and portrayal rules.
- **Symbol ontology:** defines the concept of SymbolSet and Symbol and structural definition of Symbol components.
- **Symbolizer ontology:** defines the concepts of Symbolizers defining instructions to render data as graphics.
- **Graphic Ontology:** defines graphic elements including graphic objects and attributes.

The screenshot shows the 'Create Vocabulary' form in the Vocabulary Management Services interface. The form has a dark header bar with the title 'Vocabulary Management Services', a search bar, and a 'Vocabs' dropdown. Below the header, there is a 'Create Vocabulary' button and a 'Cancel' button. The form fields are:

- Title:** Name your vocabulary
- Prefix:** Give your vocabulary a short prefix
- Category:** Select vocabulary category... (dropdown menu)
- Namespace:** Select vocabulary category... (dropdown menu)
- Description:** Select vocabulary category... (dropdown menu)

The dropdown menu for 'Category' is open, showing the following options:

- Select vocabulary category...
- Glossary
- Gazetteer
- Dictionary
- CodeList
- Thesaurus
- Taxonomy
- Ontology
- Authority File

# Style Ontology

The Style Ontology defines concept of Style, Portrayal Rule Set, Portrayal Rule, Rule Condition and PortrayalContext. [The Figure](#) below shows an overview of the model in UML.

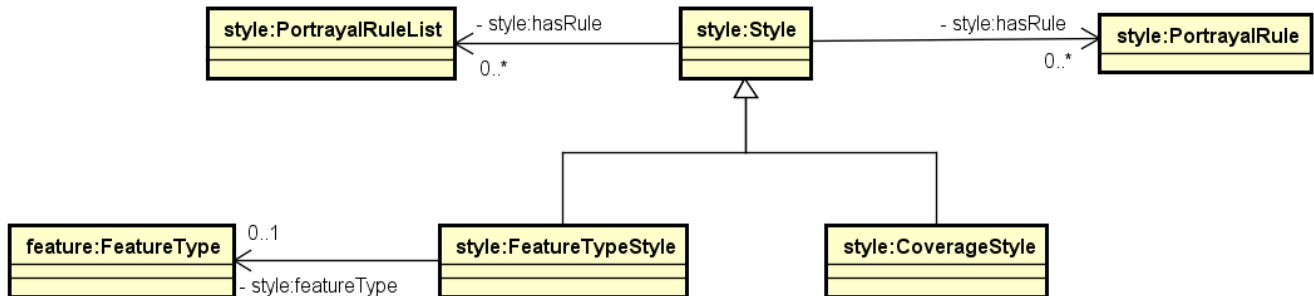


Figure 18. Style Model Overview

## Style

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

Table 69. Properties of the Style Class

Property	Usage Note	Range	Multiplicity and use
<b>name</b>	Name identifier of the style.	String	One
<b>dct:title</b>	Multilingual human-readable title for the style		1..n (one per language)
<b>dct:description</b>	Multilingual human-readable description for the style	String	1..n (one per language)
<b>hasRule</b>	PortrayalRule or ordered PortrayalRuleList associated with the Style.	Portrayal Rule or Portrayal RuleList	1..n
<b>dct:audience</b>	The intended audience of this style.	foaf:Group	0..n
<b>scope</b>	Descriptive definition of the scope of application of the style	String	0..1
<b>language</b>	Language associated with the style	String	0..n
<b>style:symbolSet</b>	SymbolSet associated with the style	SymbolSet	0..n



Property	Usage Note	Range	Multiplicity and use
<b>fieldOfApplication</b>	The field of application of this style, where values are defined as SKOS concept in a taxonomy.	skos:Concept	0..n

## FeatureTypeStyle

A **FeatureTypeStyle** is a style that is applied for a specific **feature:FeatureType**. **FeatureTypeStyle** inherits all properties from **Style**.

Table 70. Properties of the Style Class

Property	Usage Note	Range	Multiplicity and use
<b>name</b>	Name identifier of the style.	String	One
<b>dct:title</b>	Multilingual human-readable title for the style		1..n (one per language)
<b>dct:description</b>	Multilingual human-readable description for the style	String	1..n (one per language)
<b>featureType</b>	FeatureType associated with the style	1..n	<b>hasRule</b>

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

```

@prefix :      <http://www.opengis.net/testbed/12/portrayal/ems/style#> .
@prefix feature: <http://www.opengis.net/ont/feature#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ems:   <http://www.opengis.net/testbed11/ont/incident/ems#> .
@prefix style: <http://www.opengis.net/ont/portrayal/style#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix group: <http://www.socialml.org/ontologies/group#> .

:EMSIncidentStyle a      style:FeatureTypeStyle ;
                  dct:audience
<http://ows.usersmarts.com/ldapp/audiences/community/CanadianEmergencyAndDisasterManagement> ;
                  dct:description  "Style defining the set of rules for mapping incident types
from EMS to symbology" ;
                  dct:title        "EMS Incident Type Style" ;
                  style:featureType ems:EMSIncident ;
                  style:hasRule     :ems.incident.hazardousMaterial.radiologicalHazard-
portrayal-rule , :ems.incident.temperature.windChill-portrayal-rule,
:ems.incident.health-portrayal-rule , :ems.incident.hazardousMaterial-portrayal-rule
.

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

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

```

## CoverageStyle

A **CoverageStyle** is a style that is applied for a Coverage. **CoverageStyle** inherits all properties from **Style**. This concept is introduced as a placeholder for future extension.

## PortrayalRuleSet

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

Table 71. Properties of the PortrayalRuleSet Class

Property	Usage Note	Range	Multiplicity and use
<b>dct:title</b>	Multilingual human-readable name for the PortrayalRuleSet	string	0..n (one per language)
<b>dct:description</b>	Multilingual human-readable description for the PortrayalRuleSet	string	0..n (one per language)
<b>hasRule</b>	PortrayalRule member of this PortrayalRuleSet.	Portrayal Rule	0..n

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

#### EMS PortrayalRuleSet Example

```

: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 .

```

## 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**). The Table below summarizes its properties.

Table 72. Properties of the PortrayalRule

Property	Usage Note	Range	Multiplicity and use
<b>dct:title</b>	Multilingual human-readable title for the PortrayalRule	string	0..n (one per language)
<b>dct:description</b>	Multilingual human readable description for the PortrayalRule.	string	0..n (one per language)
<b>hasCondition</b>	The conditions that needs to be satisfied by the rule	Portrayal RuleCondition	0..n

Property	Usage Note	Range	Multiplicity and use
<b>maxScaleDenominator</b>	The maximum scale denominator to which the rule applies	integer	0..1
<b>minScaleDenominator</b>	The minimum scale denominator to which the rule applies	integer	0..1
<b>portrayalContext</b>	The context of application of the PortrayalRule (placeholder for future extension)	Portrayal Context	0..n
<b>symbol</b>	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 PortrayalRule Example

```

: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:hasRuleCondition :ems.incident.temperature.windChill-portrayal-rule-
condition;
  style:maxScaleDenominator "100000.0"^^xsd:double ;
  style:minScaleDenominator "0.0"^^xsd:double ;
  style:symbol
<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.temperature.windChill-
symbol> .

```

## RuleCondition

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

Table 73. Properties of the RuleCondition

Property	Usage Note	Range	Multiplicity and use
<b>dct:title</b>	Multilingual human-readable name for the RuleCondition	string	0..n (one per language)
<b>dct:description</b>	Multilingual human-readable description for the RuleCondition	string	0..n (one per language)

Property	Usage Note	Range	Multiplicity and use
<b>hasConstraint</b>	The encoding of the constraint defining the condition	Constraint	0..n

To provide an extensible mechanism to express constraints for the condition, we modified the previous ontology in Testbed 11, which uses specialized properties (sparqlCondition, rifCondition, ogcFilterCondition)

Table 74. Properties of the Constraint

Property	Usage Note	Range	Multiplicity and use
<b>constraintLanguage</b>	The constraint language used to encode the constraint. The following constraint languages URLs are currently recommended: * SPARQL: <a href="http://www.w3.org/ns/sparql-service-description#SPARQL11Query">http://www.w3.org/ns/sparql-service-description#SPARQL11Query</a> * OGC Filter: <a href="http://schemas.opengis.net/filter">http://schemas.opengis.net/filter</a> * RIF : <a href="http://www.w3.org/2007/rif">http://www.w3.org/2007/rif</a> * CQL : <a href="http://www.opengis.net/specs/cql">http://www.opengis.net/specs/cql</a>	URL	1
<b>body</b>	The constraint body expressed in the constraint language	String	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 RuleCondition Example

```
:ems.incident.temperature.windChill-portrayal-rule-condition
  a style:RuleCondition ;
  style:hasConstraint
  [
    style:Constraint ;
    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}" ;
    style:constraintLanguage <http://www.w3.org/ns/sparql-service-
description#SPARQL11Query>
  ] ;
  style:hasConstraint
  [
    a style:Constraint ;
    style:body
"ogc:Filter><ogc:PropertyIsEqualTo><ogc:PropertyName>incidentType</ogc:PropertyName><
ogc:Literal>http://www.opengis.net/taxonomy/ems#ems.incident.temperature.windChill</og
c:Literal></ogc:PropertyIsEqualTo></ogc:Filter>" ;
    style:constraintLanguage <http://schemas.opengis.net/filter>
  ] ;
  style:hasConstraint
  [
    a style:Constraint ;
    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:constraintLanguage <http://www.w3.org/2007/rif>
  ]
```

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 Constraints expressed in SPARQL and RIF 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.

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" .
```

## PortrayalRuleList and RuleItem

In some instance, rules needs to be executed in a given order with support of fallback rules. The **PortrayalRuleList** defines a ordered list of **RuleItem** instances. The PortrayalRuleList is implemented by an **rdf:List**.

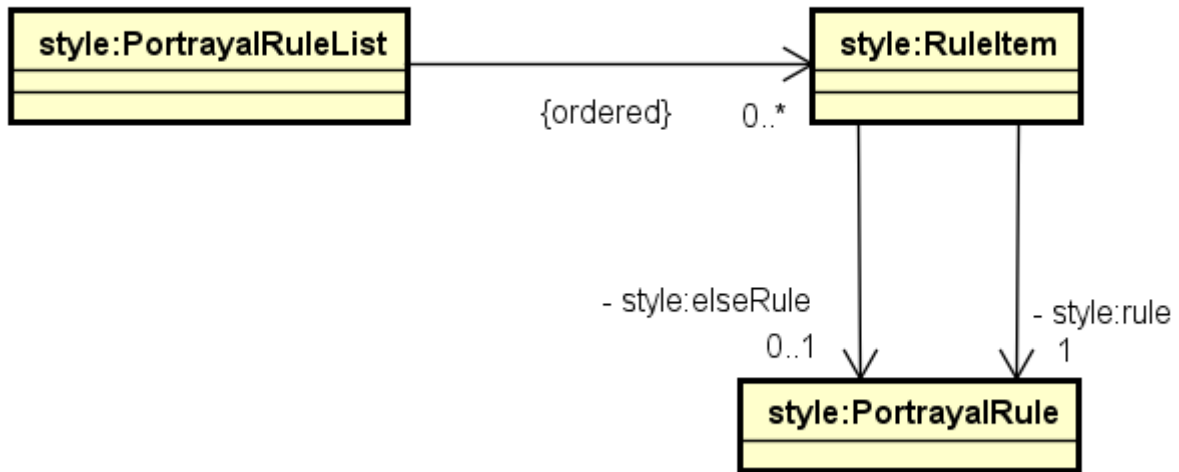


Figure 19. PortrayalRuleList

The **RuleItem** defines a placeholder referring indirectly to a portrayal rule. However it also provides a reference to fallback portrayal rules (elseRule) if the rule is not applicable. The RuleItem is used when the order of application of the rules is important. The RuleItem are used as element of **PortrayalRuleList**.

Table 75. Properties of the RuleItem\*\*

Property	Usage Note	Range	Multiplicity and use
<b>rule</b>	The rule associated with the rule item	Portrayal Rule	1

## Symbology Ontology

The Symbology Ontology is a microtheory that defines the conceptual model for defining **SymbolSet** and **Symbol** with their structural components called **Symbolizer**.

The [Figure](#) below shows an overview of the model.



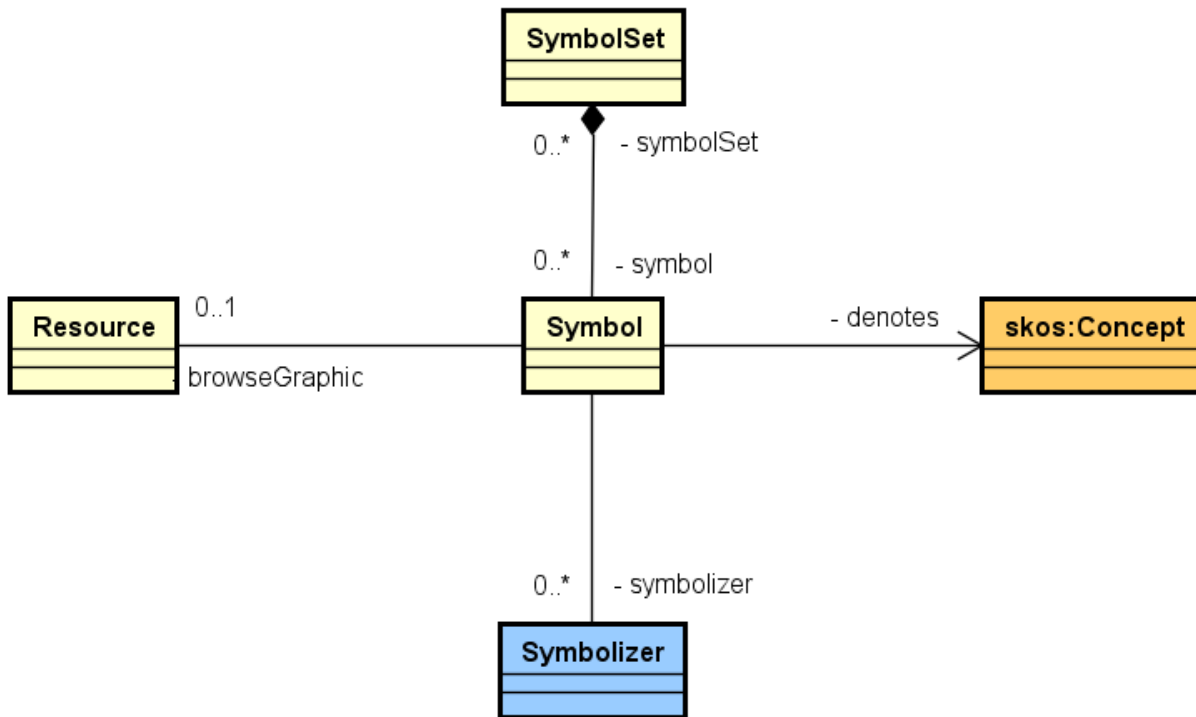


Figure 20. Symbology Model Overview

## 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. The Table below summarizes the **SymbolSet** properties.

Table 76. Properties of the SymbolSet Class

Property	Usage Note	Range	Multiplicity and use
<b>dct:identifier</b>	Unique identifier for the symbol set mainly used by machine.	string	One
<b>dct:title</b>	Multilingual human-readable title for the SymbolSet	string	0..n (one per language)
<b>dct:description</b>	Multilingual human-readable description for the SymbolSet	string	0..n (one per language)
<b>specification</b>	Cites the specification standard for the SymbolSet	Resource	0..1
<b>symbol</b>	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:symbol        :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> .

```

## Symbol

A Symbol is the type used to define symbol classes. Symbols are collected into symbol sets. A symbol has one machine readable identifier. It 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. The table below summarizes the Symbol properties.

Table 77. Properties of the Symbol Class

Property	Usage Note	Range	Multiplicity and use
<b>dct:identifier</b>	Machine readable name for the symbol. The identifier should be unique	string	1
<b>dct:title</b>	Multilingual human-readable title for the Symbol	string	0..n (one per language)
<b>dct:description</b>	Multilingual human-readable description for the Symbol	string	0..n (one per language)
<b>specification</b>	Reference to the full details of the portrayal symbol	Resource	0..1
<b>browseGraphic</b>	Reference a graphic representing the symbol	Resource	0..1
<b>denotes</b>	Concept that is denoted by the symbol	skos:Concept	0..n

Property	Usage Note	Range	Multiplicity and use
<b>symbolizer:symbolizer</b>	Defines the symbolizer rendering the symbol	symbolizer:Symbolizer	0..n
<b>symbolsSet</b>	The SymbolSet which this symbol belongs to.	SymbolSet	0..n
<b>skos:notation</b>	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 Symbol Example

```

:ems.incident.roadway.roadwayClosure-symbol
  a
    symbol:Symbol ;
  dct:title
    "roadwayClosure" ;
  symbol:browseGraphic
    <http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.roadway.roadwayClosure.png> ;
  symbol:denotes
    <http://www.opengis.net/taxonomy/ems#ems.incident.roadway.roadwayClosure> ;
  symbol:specification
    <https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf> ;
  symbol:symbolName
    "ems.incident.roadway.roadwayClosure" ;
  symbol:symbolsSet
    :EMSSymbolSet ;
  symbolizer:symbolizer
    :ems.incident.roadway.roadwayClosure-pointSymbolizer .

:ems.incident.roadway.roadwayClosure-pointSymbolizer
  a
    symbolizer:PointSymbolizer ;
  dct:title
    "PointSymbolizer for roadwayClosure symbol" ;
  graphic:graphicSymbol
    [ a
      graphic:GraphicSymbol ;
      graphic:externalGraphic
        <http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.roadway.roadwayClosure.png>
    ] .

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

```

# Symbolizer Microtheory

The Symbolizer ontology defines a set of symbolizers (also called renderers) that defines instructions for rendering data to graphic representation. The ontology is built on top of the Graphic Ontology which defines graphic objects and properties. Symbolizers are referred by **symbol:Symbol** to describe how conceptual symbol are rendered into graphics. The details of the ontology is described in Appendix C. This section describes an overview of the microtheory, examples and rationale of some of the design decisions.

The OGC Symbology Encoding defines a number of symbolizers that uses SVG parameters defines a key value pair. While this mechanism provides extensibility by accomodate future key value pairs, there is no schema allowing to know which keys are valid for a given symbolizer. We decided to move away from the approach used in the previous testbed 11, based on the ISO 19117 model which introduces symbol definition, symbol components and graphics. We found out that the ISO 19117 was not adequate to represented as semantic descriptive model. The ISO 19117 is more geared toward an implementation which can use function to calculate positioning of symbol components. We decided to align out model more toward main stream map renderer and descriptive standards such as SVG. ===== Symbolizer Hierarchy

The top concept of the Symbolizer ontology is **Symbolizer**. A **Symbolizer** describes how a feature is to appear on a map. The Symbolizer describes not just the shape that should appear but also how visual variables are defined using graphical properties such as color and opacity. A Symbolizer is obtained by specifying one of subclass of Symbolizer and then supplying parameters to override its default behavior. The hierarchy of symbolizer is illustrated in the following figure.

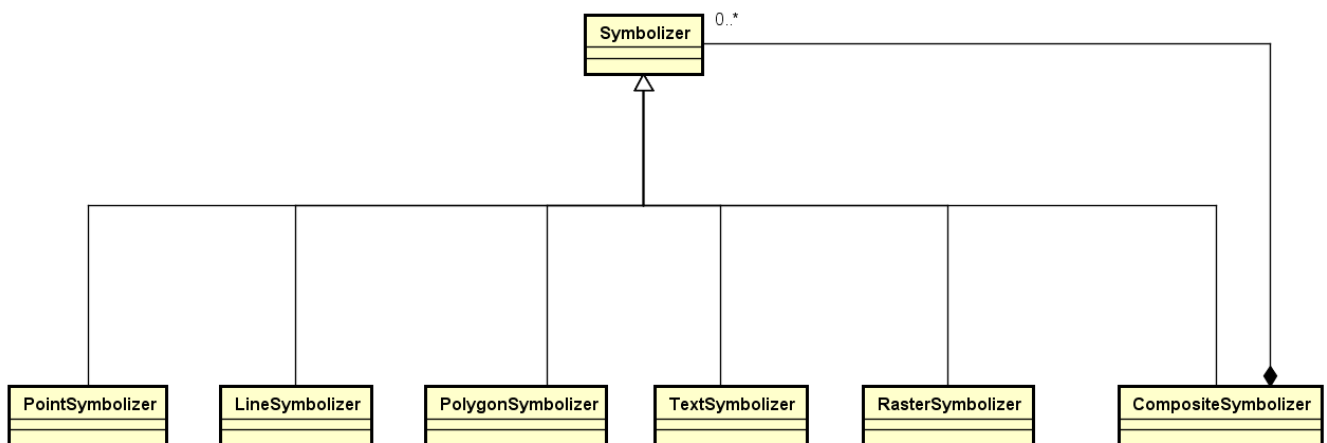


Figure 21. Symbolizer Hierarchy

Note that the hierarchy uses similar terminology than the OGC SE specification, except that it introduces an additional class called **CompositeSymbolizer** that defines a composition of symbolizers applied in a given order and using a composition operation (default is source over). This version of the ontology mostly focused on rendering of feature (vector) data, and is not addressing in depth the Raster Symbolizer, which is introduced as a placeholder for future extension. Other extensions of symbolizer may be defined down the road for more specialized use-cases such as complex symbologies that are difficult to express with graphic descriptors (example MIL 2525 symbology).

## Symbolizer

Symbolizer is the top class of all the symbolizer and provides properties that are inherited by all subclasses.

Table 78. Properties of the Symbolizer Class

Property	Usage Note	Range	Multiplicity and use
name	symbolizer name	xsd:string	0..1
dct:title	title of the symbolizer	xsd:string	0..1
dct:description	description of the symbolizer	xsd:string	0..1
:uom	Unit of measure that applies to all elements included inside a Symbolizer such as stroke-width, Size, fontsize, Gap, InitialGap, Displacement and PerpendicularOffset. If no uom is set inside of Symbolizer, all units are measured in pixel.	URI	0..1

## Point Symbolizer

A **PointSymbolizer** is defined as a Symbolizer that is used to draw a “graphic” at a point. If the geometry associated with the **PointSymbolizer** is not a point, such as a line, polygon, raster, a representative point of the geometry (typically centroid) should be used.

Table 79. Properties of the PointSymbolizer Class

Property	Usage Note	Range	Multiplicity and use
graphic:graphicSymbol	graphic symbol	graphic:GraphicSymbol	0..1
:geometryProperty	Geometric Property associated with the point symbolizer	:Property	0..n

## Line Symbolizer

A **LineSymbolizer** is a **Symbolizer** that is used to render a "stroke" along a linear geometry type such as string of line segments. Geometry types other than inherently linear types can also be used. If a point geometry is used, it should be interpreted as a line of “epsilon” (arbitrarily small) length with a horizontal orientation centered on the point, and should be rendered with two end caps. If a "area" geometry (for example polygon) is used, then its closed outline is used as the line string (with no end caps). If a raster geometry is used, its coverage-area outline is used for the line, rendered with no end caps.

Table 80. Properties of the LineSymbolizer Class

Property	Usage Note	Range	Multiplicity and use
:geometry Property	Geometric Property associated with the symbolizer	:Property	0..n
graphic:stroke	Stroke associated with the line symbolizer	graphic:Stroke	0..1
graphic:perpendicularOffset	Perpendicular offset	xsd:decimal	0..1

## Polygon Symbolizer

A **PolygonSymbolizer** is a **Symbolizer** that is used to render the area enclosed by a polygon. If a polygon has “holes,” then they are not filled, but the borders around the holes are stroked in the usual way. “Islands” within holes are filled and stroked, and so on. If a point geometry is referenced instead of a polygon, then a small, square, ortho-normal polygon should be constructed for rendering. If a line is referenced, then the line (string) is closed for filling (only) by connecting its end point to its start point, any line crossings are corrected in some way, and only the original line is stroked. If a raster geometry is used, then the raster-coverage area is used as the polygon.

Table 81. Properties of the PolygonSymbolizer Class

Property	Usage Note	Range	Multiplicity and use
:geometry Property	Geometric Property associated with the symbolizer	:Property	0..n
graphic:stroke	Stroke associated with the polygon symbolizer	graphic:Stroke	0..1
graphic:fill	Fill associated with the polygon symbolizer	graphic:Fill	0..1
graphic:perpendicularOffset	Perpendicular offset	xsd:decimal	0..1
graphic:displacement	Displacement	graphic:Translation	0..1

## Text Symbolizer

A **TextSymbolizer** is a **Symbolizer** that is used to render text.

Table 82. Properties of the TextSymbolizer Class

Property	Usage Note	Range	Multiplicity and use
:geometry Property	Geometric Property associated with the symbolizer	:Property	0..n
graphic:font	Font associated with the text symbolizer	graphic:Font	0..1
graphic:fill	Fill associated with the text symbolizer	graphic:Fill	0..1
graphic:halo	Halo associated with the text symbolizer	graphic:Halo	0..1
graphic:labelPlacement	LabelPlacement associated with the text symbolizer (could be PointPlacment or LinePlacement for example)	graphic:LabelPlacement	0..1

## Raster Symbolizer

A **RasterSymbolizer** is a **Symbolizer** that is used to render raster. This symbolizer has been introduced as a placeholder for future extensions, as this testbed was primarily focused on symbolization of vector data.

## Composite Symbolizer

A **CompositeSymbolizer** is a symbolizer composed of a list of Symbolizers that applied in a given order using a given composition operation. Default composition operation is **src-over** composition.

Table 83. Properties of the CompositeSymbolizer Class

Property	Usage Note	Range	Multiplicity and use
:comp-op	Composition operator valid values are "src-over", "dest-out", "dest-over"	xsd:string	0..1
:items	List of Symbolizer rendered in the order of the list	rdf:List of Symbolizers	1

# Graphics Microtheory

## Context

This document is prepared in the context of the Semantic Portrayal Service - a true semantic-enabled service that demonstrates semantic integration and interoperability across disparate portrayal catalogues.

## Scope

The objective of this ontology is to define vocabulary terms describing graphic elements (objects and properties) using Linked Data encoding. The use of ontology language such as OWL provides a powerful mechanism to express the semantic, classification, relationships and constraints of the graphic elements. The Linked data encoding for the graphics enables to build a "Web of Graphics" that could be referenced by other resources and the decentralized distribution of graphics information on the web, favoring reusability of graphics and linkage to application that can go beyond styling application. In the case of map portrayal application, the graphic ontology is used by symbolizers to instruct renderers the way to render features on a map.

To model the graphic elements of the ontology, we try to use a model that is close to the way graphic properties are defined in CSS and SVG. For example to encode color, the preferred way is to use literal with a **CSSColorLiteral** datatype that uses the same syntax than CSS and SVG. Similarly length can be expressed using different unit of measures (percent, cm, mm). Instead of introducing a concept of Length and Unit of Measure that could be very verbose, we choose to use introduce a datatype **LengthType** that is based on the SVG and CSS syntax.

## Terminology used in the Graphics Ontology

The Graphics Ontology reuses terms from various existing specifications. The default namespace used in the documentation of the ontology is <http://www.opengis.net/ont/portrayal/graphic#>. The preferred prefix for this namespace is **graphic**. Classes and properties in the next sections that have been taken from the following namespaces.

Table 84. Namespace Prefixes

Prefix	Namespace
dct	<a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a>
geosparql	<a href="http://www.opengis.net/ont/geosparql#">http://www.opengis.net/ont/geosparql#</a>
owl	<a href="http://www.w3.org/2002/07/owl#">http://www.w3.org/2002/07/owl#</a>
rdf	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>
rdfs	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>
xsd	<a href="http://www.w3.org/2001/XMLSchema#">http://www.w3.org/2001/XMLSchema#</a>

## Graphics Ontology Classes

This figure shows a UML Diagram of all classes and properties included in the Graphics Ontology.



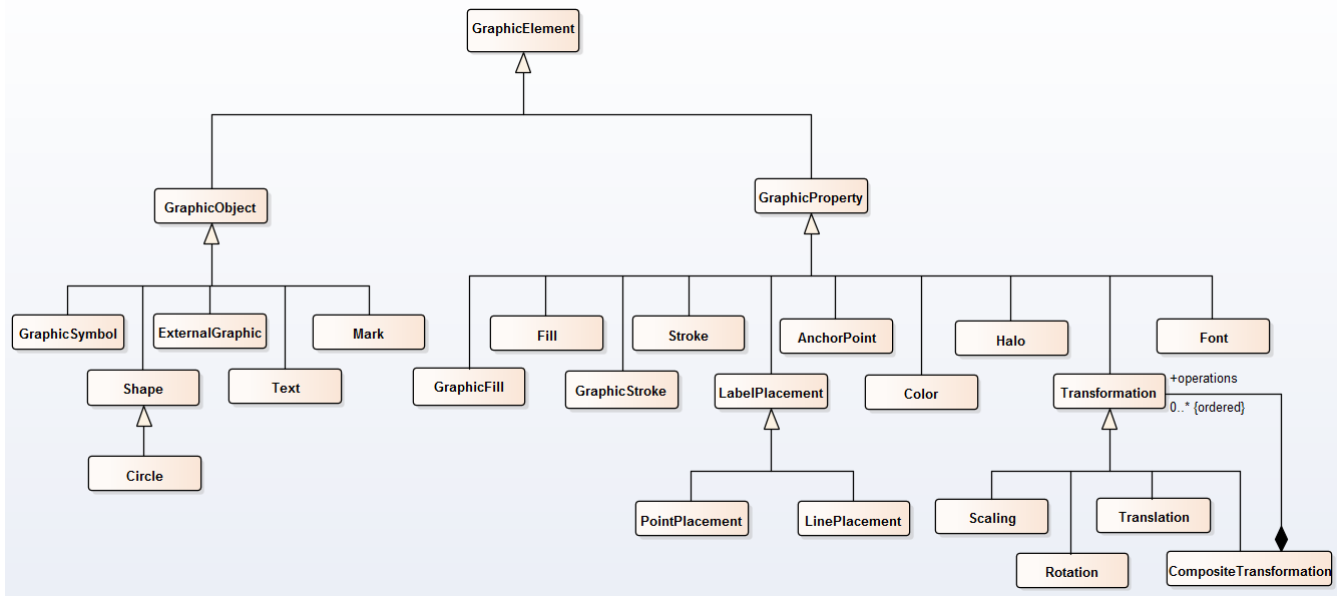


Figure 22. Graphics Ontology

### Graphic Element

The **:GraphicElement** concept is the top concept of the Graphic Ontology. It is aligned with the definition introduced in ISO 19117. **:GraphicElement** is the abstract root for the graphic elements, as defined in a graphic specification language (such as SVG or OGC SE), that defines a symbol. The graphic elements may be **graphic objects**, such as shapes, texts or **graphic properties** such as colour, stroke, fill, font. A **:GraphicElement** has two subclasses: **:GraphicObject** and **:GraphicProperty**.

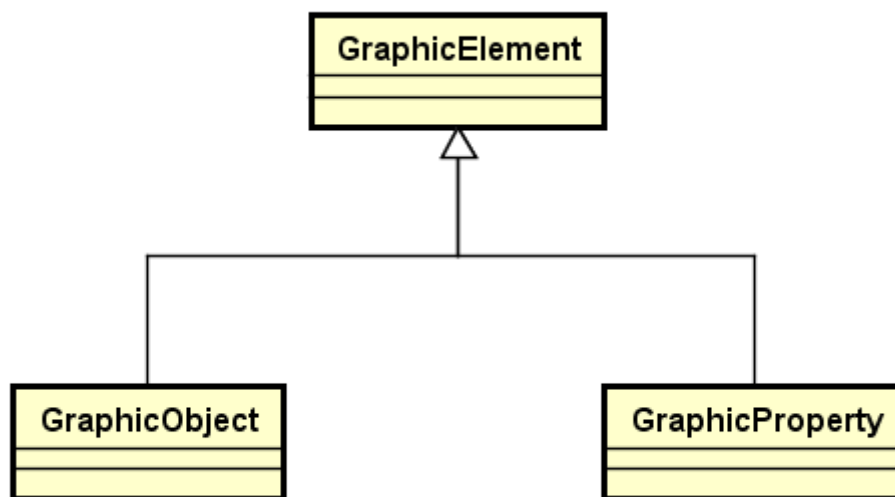


Figure 23. Graphic Element Hierarchy

### Graphic Object

A **:GraphicObject** is an abstract class defined as a **:GraphicElement** that can be drawn such as a graphic shape (oval, ellipse, rectangle, path) , Raster, Glyphs (text and graphic symbols) .

**NOTE**

The term **SY\_GraphicObject** is defined in ISO 19117 as "an abstract specialization of **SY\_GraphicElement** as a graphic object defined in a graphic specification language. Graphic objects are graphic elements, such as ovals, rectangles, or paths. A graphic object in turn has properties, such as location attributes, size attributes, colour attributes, etc."

The **:GraphicObject** is the base class of a hierarchy illustrated below. The detail of each subclass is described in next sections.

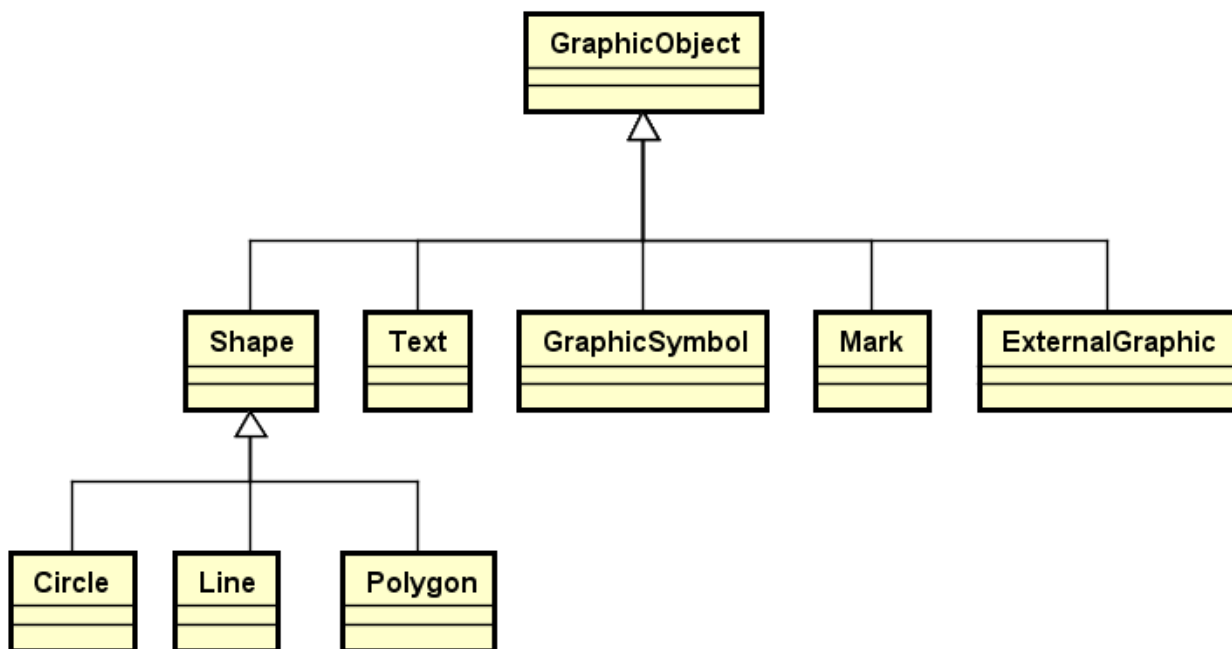


Figure 24. Graphic Object Hierarchy

### Shape

A **:Shape** is a **:GraphicObject** defining a geometric figure.

**NOTE**

The term **:Shape** is defined in SVG as an element defined by some combination of straight lines and curves. Specifically: **path**, **rect**, **circle**, **ellipse**, **line**, **polyline**, and **polygon**.

The description of the geometry of the shape can be defined in different geometry description language such as GML, WKT and SVG. The ontology reuses the definition of GeoSPARQL **geosparql:asWKT**, **geosparql:asGML** to represent respectively geometry encoded in WKT and GML. GeoSPARQL introduces two datatypes to validate the encoding (**geosparql:WKTLiteral** and **geosparql:GMLLiteral**). To accommodate more complex path, the graphic ontology introduces an additional property (**asSVGPath**) and datatype for encoding SVG Path (**:svgPathLiteral**), which uses the syntax of path defined in SVG.

Table 85. Properties of the Shape Class

Property	Usage Note	Range	Multiplicity and use
<b>geosparql:asWKT</b>	Define the geometry using Well-Known-Text Encoding	geosparql:wktLiteral	0..1 (optional)
<b>geosparql:asGML</b>	Define the geometry using GML Geometry Encoding	geosparql:gmlLiteral	0..1 (optional)
<b>svgPath</b>	specifies a path of the shape using SVG graphic language	:svgPathLiteral	0..1 (optional)

## Graphic Symbol

A **:GraphicSymbol** is a **:GraphicObject** with an inherent shape, color(s), and possibly size. A **:GraphicSymbol** can be very informally defined as “a little picture” and can be of either a raster or vector-graphic source type. The term “GraphicSymbol” is used since the term “symbol” is similar to “Symbolizer” which is used in a different context in SE.”

### NOTE

The term **:GraphicSymbol** is synonymous to **Graphic** in the SE Implementation Specification, which is defined as a “graphic symbol” with an inherent shape, color(s), and possibly size. A “graphic” can be very informally defined as “a little picture” and can be of either a raster or vector-graphic source type. The term “graphic” is used since the term “symbol” is similar to “Symbolizer” which is used in a different context in SE.”

Table 86. Properties of the Graphic Symbol Class

Property	Usage Note	Range	Multiplicity and use
<b>:externalGraphic</b>	external graphics contained a graphic symbol	:ExternalGraphic	0..1 (optional)
<b>:mark</b>	Marks contained in a graphic symbol	:Mark	0..1 (optional)
<b>:opacity</b>	opacity of the color or the content the current object is filled with	xsd:decimal (0.0 to 1.0), representing percent opacity	0..1 (optional)
<b>:size</b>	Absolute size	xsd:decimal, Non-negative real number in pixels	0..1 (optional)

Property	Usage Note	Range	Multiplicity and use
<b>:anchorPoint</b>	The location inside object to use for anchoring to main geometry point	AnchorPoint	0..1 (optional)

## External Graphic

An **ExternalGraphic** gives a reference to an external raster or vector graphical object. It allows a reference to be made to an external graphic file with a Web URL or to in-line content. The **:onlineResource** property gives the URL and the **:format** property identifies the expected document MIME type of a successful fetch. Knowing the MIME type in advance allows the styler to select the best-supported format from the list of URLs with equivalent content. The **:inlineContent** property allows the content of an external graphic object to be included in-line as a `xsd:base64Binary` datatype.

**NOTE** The term **ExternalGraphic** is defined in the SE Implementation Specification as an element that "allows a reference to be made to an external graphic file with a Web URL or to in-line content."

Table 87. Properties of the ExternalGraphic Class

Property	Usage Note	Range	Multiplicity and use
<b>:onlineResource</b>	URL of external graphic	URL	0..1 (conditional)
<b>:inlineContent</b>	Content of external graphic	<code>xsd:base64Binary</code>	0..1 (conditional)
<b>:format</b>	MIME type of external graphic	<code>xsd:string</code>	1 (mandatory)

## Mark

A **:Mark** is a **:GraphicObject** that describes a graphic symbol based on a shape with coloring applied to it.

**NOTE** The term **:Mark** is defined in SE as an element that defines a "shape" which has coloring applied to it. The Mark element serves two purposes. It allows the selection of simple shapes, and, in combination with the capability to select and mix multiple external-URL graphics and marks, it allows a style to be specified that can produce a usable result in a best-effort rendering environment, provided that a simple Mark is included at the bottom of the list of sources for every Graphic.

Table 88. Properties of the Mark Class

Property	Usage Note	Range	Multiplicity and use
<b>:fill</b>	Mark fill	:Fill	0..1 (optional)

Property	Usage Note	Range	Multiplicity and use
<b>:stroke</b>	Marks contained in a graphic symbol	:Mark	0..1 (optional)
<b>:shape</b>	The shape associated with the Mark	:Shape	0..1 (conditional)
<b>:onlineResource</b>	URL of the graphic symbol	URI	0..1 (conditional)
<b>:inlineContent</b>	Content of the graphic symbol	xsd:base64Binary	0..1 (conditional)
<b>:format</b>	format or MIME type	xsd:string	0..1 (conditional)
<b>:markIndex</b>	Index to an individual mark in a mark archive; include when needed to uniquely identify mark.	xsd:string	0..1 (optional)

## Text

A **:Text** is a **:GraphicObject** that describes a graphic symbol based on a shape with coloring applied to it.

### NOTE

The term **:Text** is defined in SVG as a "graphic element consisting of text. The attributes and properties on the text element indicate such things as the writing direction, font specification and painting attributes which describe how exactly to render the characters."

Table 89. Properties of the Text Class

Property	Usage Note	Range	Multiplicity and use
<b>:font</b>	The font information for the label	:Font	0..1 (optional)
<b>:graphicSymbol</b>	A graphic symbol to be displayed behind the label text	:GraphicSymbol	0..1 (optional)
<b>:textLabel</b>	The text content for the label	xsd:string	0..1 (optional)
<b>:labelPlacement</b>	sets the position of the label relative to its associated geometry.	:LabelPlacement	0..1 (conditional)
<b>:halo</b>	Creates a colored background around the label text, for improved legibility.	:cssColorLiteral	0..1 (optional)
<b>:fill</b>	The fill style of the label text	:Fill	0..1 (optional)

## Graphic Datatypes

The Graphic ontology introduces a couple of datatypes that are used to indicate RDF processor how to convert and validate string literal to internal value.

## Graphic Properties

### GraphicProperty

A **:GraphicProperty** is defined as a **:GraphicElement** that represents attributes that are used to modify the appearance of the graphic objects, such as its size, orientation, color, outline stroke, fill pattern, fonts, font size. While in several standards such as ISO 19117 and OGC SE, graphic properties are encoded as name/value pairs, which provide flexibility to add future extensions with new names, they do not provide any semantic to the name and make hard to define the adequate restrictions of graphic properties on specific graphic objects. It is also prone to misinterpretation by implementers.

The use of ontology provides a clear semantic for each property and relationships to other properties. For example, **:font-color**, **:stroke-color** and **:fill-color** are defined as subproperties of **:color**. The graphic ontology introduces two ways to model graphic properties. The first one is by defining subproperties of the datatype property **:graphicProperty** (for example **:stroke-width**, **:stroke-color**, **:font-size**). The second way is by defining a subclass of the **:GraphicProperty** such as **:Stroke**, **:Color**, **:Font**, **:Fill**, which are often used as reusable containers for of multiple graphic properties.

#### NOTE

The term **GraphicProperty** is defined in ISO 19117 as "a templatized specialization of **SY\_GraphicElement** used to define graphic properties, such as location attributes, size attributes, colour attributes, etc. As such it shall implement all inherited attributes, operations and associations. "

### AnchorPoint

A **:AnchorPoint** gives the locaiton inside of a Graphic Object to use for anchor the graphic object to the main-geometry point. The coordinates are give as two floating-point numbers in th **AnchorPointX** and **AnchorPointY** elements each with values between 0.0 and 1.0 inclusive. The default point is X=0.5, Y=0.5, which is at the middle height and middle length of the graphic/label text.

Table 90. Properties of the AnchorPoint Class

Property	Usage Note	Range	Multiplicity and use
<b>:anchorPointX</b>	The x-coordinate location inside object to use for anchoring to main geometry point	AnchorPoint	0..1 (optional)
<b>:anchorPointY</b>	The y-coordinate location inside object to use for anchoring to main geometry point	AnchorPoint	0..1 (optional)

### Stroke

A **:Stroke** is a **:GraphicProperty** that describes a linear stroke. The **:GraphicStroke** element describes graphics repeated linearly and is described below. There are three basic types of strokes:

solid-color, **GraphicFill** (stipple), and repeated linear **GraphicStroke**. A repeated linear graphic is plotted linearly and has its graphic Symbolizer bent around the curves of the line string, and a graphic fill has the pixels of the line rendered with a repeating area-fill pattern. If neither a **GraphicFill** nor **GraphicStroke** element is given, then the **LineStyle** will render a solid color.

**NOTE**

The term **:Stroke** is defined in SE as a "graphical element of the LineSymbolizer that encapsulates the graphical-Symbolization parameters for linear geometries."

Table 91. Properties of the Stroke Class

Property	Usage Note	Range	Multiplicity and use
<b>:graphicStroke</b>	A graphic to be used instead of a stroke	:GraphicStroke	0..n (optional)
<b>:graphicFill</b>	A graphic to be repeated in the area instead of a solid fill color	:GraphicFill	0..1 (optional)
<b>:stroke-color</b>	Color	xsd:cssColorLiteral	0..1 (optional)
<b>:hasStrokeColor</b>	Color	:Color	0..1 (optional)
<b>:stroke-opacity</b>	Opacity	xsd:decimal, 0.0 to 1.0, representing percent opacity	0..1 (optional)
<b>:stroke-width</b>	Width	xsd:decimal, Non-negative real number in pixels.	0..1 (optional)
<b>:stroke-linecap</b>	Linecap	xsd:string, either "butt", "round", or "square"	0..1 (optional)
<b>:stroke-linejoin</b>	Linejoin	:Linejoin, either "miter", "round", or "bevel"	0..1 (optional)

Property	Usage Note	Range	Multiplicity and use
<b>:stroke-dasharray</b>	Dash array – a sequence of distances (dash, space, dash, etc.) to draw	xsd:string of separated non-negative real numbers in pixels	0..1 (optional)
<b>:stroke-dashoffset</b>	Dash offset – the distance into the dasharray at which to start drawing	xsd:decimal, Non-negative real number in pixels	0..1 (optional)

### GraphicStroke

A **:GraphicStroke** is a **:GraphicProperty** that describes a graphic symbol repeated linearly. It contains a Graphic symbol and may contain an InitialGap and/or Gap if relevant. The **:graphicSymbol** specifies the linear **:GraphicSymbol**. Proper stroking with a linear graphic symbol requires two “hot-spot” points within the space of the graphic symbol to indicate where the rendering line starts and stops. In the case of raster images with no special mark-up, this line will be assumed to be middle pixel row of the image, starting from the first pixel column and ending at the last pixel column. **GraphicStrokes** should not be used for **Strokes** that are part of Marks

**NOTE** The term **:GraphicStroke** is defined in SE with the same definition.

Table 92. Properties of the GraphicStroke Class

Property	Usage Note	Range	Multiplicity and use
<b>:graphicSymbol</b>	The linear graphic symbol to be drawn	:GraphicSymbol	1..n (mandatory)
<b>:initialGap</b>	How far away the first graphic symbol will be drawn relative to the start of the rendering line	xsd:decimal, Non-negative real number in pixels	0..n (optional)
<b>:gap</b>	Distance between two graphic symbols	xsd:decimal, Non-negative real number in pixels	0..n (optional))



## Fill

A **:Fill** is a **:GraphicProperty** that describes a graphic symbol repeated linearly. It contains a Graphic symbol and may contain an InitialGap and/or Gap if relevant. The **:hasGraphicSymbol** specifies the linear **:GraphicSymbol**. Proper stroking with a linear graphic symbol requires two “hot-spot” points within the space of the graphic symbol to indicate where the rendering line starts and stops. In the case of raster images with no special mark-up, this line will be assumed to be middle pixel row of the image, starting from the first pixel column and ending at the last pixel column. **GraphicStrokes** should not be used for **Strokes** that are part of Marks

**NOTE** The term **:Fill** is defined in SE with the same definition.

Table 93. Properties of the Fill Class

Property	Usage Note	Range	Multiplicity and use
<b>:hasFillColor</b>	Color	:Color	0..1 (optional)
<b>:fill-color</b>	Color	:cssColor Literal	0..1 (optional)
<b>:fill-opacity</b>	Opacity	xsd:decimal, 0.0 to 1.0, representing percent opacity	0..1 (optional)

## GraphicFill

A **:GraphicFill** is a **:GraphicProperty** that describes the graphic that will be used to fill the area of a polygon. GraphicFills should not be used for Strokes that are part of Marks.

**NOTE** The term **:GraphicFill** is defined in SE with the same definition.

Table 94. Properties of the GraphicFill Class

Property	Usage Note	Range	Multiplicity and use
<b>:graphicSymbol</b>	The linear graphic symbol to be drawn.	:GraphicSymbol	1..n (mandatory)

## Font

A **:Font** is a **:GraphicProperty** used to represent text or symbols.

**NOTE** The term **:Font** is defined in SE as an element that "identifies a font of a certain family, style, and size". Four types of **SvgParameter** are allowed, "**font-family**", "**font-style**", "**font-weight**", and "**font-size**".

Table 95. Properties of the Font Class

Property	Usage Note	Range	Multiplicity and use
<b>:font-family</b>	gives the family name of a font to use	:FontFamily	0..n (optional)
<b>:font-style</b>	gives the style to use for a font.	xsd:string, with the allowed values being "normal", "italic", and "oblique"	0..1 (optional)
<b>:font-size</b>	gives the size to use for the font in pixels	xsd:decimal	0..1 (optional)
<b>:font-weight</b>	gives the amount of weight or boldness to use for a font	xsd:string, with allowed values being "normal" or "bold"	0..1 (optional)
<b>:hasFillColor</b>	Color	:Color	0..1 (optional)
<b>:fill-color</b>	Color	:cssColorLiteral	0..1 (optional)

### LabelPlacement

The **:LabelPlacement** element is used to position a label relative to a graphic object. It is superclass of **:PointPlacement** and **:LinePlacement**.

### LinePlacement

Table 96. Properties of the LinePlacement Class

Property	Usage Note	Range	Multiplicity and use
<b>:initialGap</b>	How far away the first label will be drawn relative to the start of the rendering line	xsd:decimal, Non-negative real number in pixels	0..1 (optional)
<b>:gap</b>	Distance between two labels	xsd:decimal, Non-negative real number in pixels	0..1 (optional))
<b>:perpendicularOffset</b>	gives the perpendicular distance away from a line to draw a label	:uoms	0..1 (optional))
<b>:generalizeLine</b>	The boolean generalizeLine property allows the actual geometry, be it a linestring or polygon to be generalized for label placement. This is for example useful for labeling polygons inside their interior when there is need for the label to resemble the shape of polygon. (See OGC Symbology Encoding)	xsd:boolean	0..1 (optional))
<b>:isAligned</b>	boolean indicating is the label is aligned with the line	xsd:boolean	0..1 (optional))
<b>:isRepeated</b>	If IsRepeated is 'true', the label will be repeatedly drawn along the line with InitialGap and Gap defining the spaces at the beginning and between labels.	xsd:boolean	0..1 )

## PointPlacement

Table 97. Properties of the PointPlacement Class

Property	Usage Note	Range	Multiplicity and use
<b>:rotation</b>	Clockwise rotation about the graphic object's anchor point	xsd:decimal (real number in degrees)	0..1 (optional)
<b>:anchorPointX</b>	The x-component of the location inside object to use for anchoring to main geometry point	xsd:decimal	0..1 (optional)
<b>:anchorPointY</b>	The y-component of the location inside object to use for anchoring to main geometry point	xsd:decimal	0..1 (optional)
<b>:displacementX</b>	The x-component of the displacement from "hot-spot" point	xsd:decimal	0..1 (optional)

## Halo

A **:Halo** is a type of **Fill** that is applied to the backgrounds of font glyphs. The use of halos greatly improves the readability of text labels. .Properties of the Halo Class

Property	Usage Note	Range	Multiplicity and use
<b>:fill</b>	Halo fill	:Fill	0..1 (optional)
<b>:radius</b>	ves the absolute size of a halo radius in pixels encoded as a floating-point number.	xsd:decimal	0..1 (optional)

## Transformation

A **:Transformation** is a **:GraphicProperty** used to apply a type of transformation, such as scaling, translation, or rotation, to a graphic object.

### Size

The **Size** element gives the absolute size of the graphic object in uoms encoded as a floating-point number. The default size for an object is context-dependent. Negative values are not allowed. The default size of an image format (such as GIF) is the inherent size of the image. The default size of a format without an inherent size (such as SVG which are not specially marked) is defined to be 16 pixels in height and the corresponding aspect in width. If a size is specified, the height of the graphic object will be scaled to that size and the corresponding aspect will be used for the width. An expected common use case will be for image graphics to be on the order of 200 pixels in linear size and to be scaled to lower sizes. On systems that can resample these graphic objects “smoothly,” the results will be visually pleasing.

Table 98. Properties of the Scale Class

Property	Usage Note	Range	Multiplicity and use
<b>:scaleX</b>	Scale factor on the X-axis direction." The default value is 0.0	xsd:decimal	1
<b>:scaleY</b>	Scale factor on the Y-axis direction." The default value is 0.0	xsd:decimal	1

### Rotation

The **Rotation** element is a basic transformation that gives the rotation of a graphic object in the clockwise direction about its center point in decimal degrees, encoded as a floating-point number. Negative values mean counter-clockwise rotation. The default value is 0.0 (no rotation). Note that there is no connection between source geometry types and rotations; the point used for plotting has no inherent direction. Also, the point within the graphic object about which it is rotated is format dependent. If a format does not include an inherent rotation point, then the point of rotation should be the centroid.

Table 99. Properties of the Rotation Class

Property	Usage Note	Range	Multiplicity and use
<b>:rotation</b>	Rotation angle in the clockwise direction about its center point in decimal degrees, encoded as a floating-point number. Negative values mean counter-clockwise rotation. The default value is 0.0 (no rotation).	xsd:decimal	1

### *Translation*

The **Translation** gives the X and Y displacements from the “hot-spot” point. This element may be used to avoid over-plotting of multiple graphic objects used as part of the same point symbol. The displacements are in units of measure above and to the right of the point. The default displacement is X=0, Y=0. If Displacement is used in conjunction with Size and/or Rotation then the graphic object shall be scaled and/or rotated before it is displaced. The term displacement, in this case, is synonymous to **translation**.

*Table 100. Properties of the Translation Class*

Property	Usage Note	Range	Multiplicity and use
<b>:displacementX</b>	Displacement on the X-axis direction." The default value is 0.0	xsd:decimal	1
<b>:displacementY</b>	Displacement on the Y-axis direction." The default value is 0.0	xsd:decimal	1

### *Composite Transformation*

A **composite transformation** is two or more transformations performed one after the other in order of the list.

*Table 101. Properties of the Composite Transformation Class*

Property	Usage Note	Range	Multiplicity and use
<b>:operations</b>	Transformation operation list executed in order of the list	rdf:List of Transformation instance	1

# Appendix D: Semantic Registry Service REST API v0.1

## Overview

This document describes the initial version of RESTful Semantic Registry Service API. We anticipate that a variety of clients may be using the Semantic Registry Service, and as a consequence it is difficult to accommodate the needs of every type of client. It is almost certain that the REST API will evolve and be modified as more requirements and features are added to the service. The hypermedia-driven API provides a robust approach to evolve the API without breaking the client ecosystem, as long as the clients are using the semantics of the link relation types. For this reason, the service has adopted a hypermedia-driven RESTful API by default, which meets level 3 of the [Richardson Maturity Model](#).

The default serialization of the information model in the service is JSON, as it is understood by most programming languages. However, to support machine-processable information, we enforce the JSON to be compatible with Linked data by using JSON-LD Context. In this way, data can be converted to Linked Data Representation and be reasoned on and linked to other information expressed as Linked Data. The REST API does support content negotiation to return a response in Linked Data Format (RDF/XML, Turtle, NTriple) when applicable (Register and Item at this point of time). This REST API provides the core minimum functionalities that are based on the **Semantic Registry Information Model (SRIM)** Application Profile.

The Semantic Registry Service provides a set of core resources (Registers, Items, SPARQL endpoint). This set of core resources has been extended to support a **Harvester service** that enables the population of the registry by collecting information from multiple sources such as OGC Catalog Services, CKAN, or Web Accessible Folder (WAF).

### NOTE

The example URL used in this documentation uses the baseURL <http://localhost:8080>. This baseURL should be replaced by the entry point (baseURL) of the service that you want to access in the format: <http://{hostname}:{port}/{rootPath}>

## HTTP Verbs

The RESTful API tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs. The following HTTP verbs are used by the Service.

Verb	Usage
GET	Used to retrieve a resource
POST	Used to create a new resource
PUT	Used to update an existing resource
DELETE	Used to delete an existing resource

## HTTP Status Codes

The Service REST API tries to adhere as closely as possible to the standard HTTP and REST conventions in its use of HTTP status codes. The following table summarizes the status codes and usage.

Status code	Usage
200 OK	The request completed successfully
201 Created	A new resource has been created successfully. The resource's URI is available from the response's <code>Location</code> header
204 No Content	An update to an existing resource has been applied successfully
400 Bad Request	The request was malformed. The response body will include an error providing further information
404 Not Found	The requested resource did not exist
405 Method Not Allowed	A request was made of a resource using a method not supported by that resource; for example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.
409 Conflict	The request could not be completed due to a conflict with the current state of the resource. This code is only allowed in situations where it is expected that the user might be able to resolve the conflict and resubmit the request.
415 Unsupported Media Type	The server is refusing to service the request because the method on the requested resource does not accept a request in that format.
500 Internal Server Error	The web server encountered an unexpected condition that prevented it from fulfilling the request.

## Headers

Every response has the following header:

Name	Description
<code>Content-Type</code>	The Content-Type of the payload, e.g. <code>application/hal+json</code>

Future headers may be added for managing access control to the resources managed by the service.

## Errors

Whenever an error response (status code  $\geq 400$ ) is returned, the body will contain a JSON object that describes the problem. The error object has the following structure:

Path	Type	Description
<code>status</code>	Number	The HTTP status code, e.g. <code>400</code>
<code>error</code>	String	The HTTP error that occurred, e.g. <code>Bad Request</code>

Path	Type	Description
message	String	A description of the cause of the error
path	String	The path to which the request was made
timestamp	String	The date and time at which the error occurred

For example, a request that attempts to apply a non-existent resource to a register item will produce the **400 Bad Request** response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Content-Length: 189

{
  "status" : 400,
  "error" : "Bad Request",
  "message" : "The resource item 'http://localhost:8080/items/123' does not exist",
  "path" : "/items",
  "timestamp" : "2016-09-30T18:37:25.119+0000"
}
```

## Paging and Sorting

### Paging

Rather than returning everything from a large result set, the REST API recognizes some URL parameters that will influence the page size (**size** parameter), the starting page number (**page** parameter), and the sorting of the result set (**sort** parameter).

See the following example, where we set the page size to 5 and request the third page (page 2) as page numbers are **zero-indexed**:

```
GET items?page=2&size=5 HTTP/1.1
Host: localhost
```

The request returns the following paginated results in HAL format:



```

{
  "_embedded": {
    ...data...
  },
  "_links": {
    "self": {
      "href": "http://localhost:8080/items"
    },
    "first": {
      "href": "http://localhost:8080/items?page=0&size=5"
    },
    "prev": {
      "href": "http://localhost:8080/items?page=1&size=5"
    },
    "next": {
      "href": "http://localhost:8080/items?page=3&size=5"
    },
    "last": {
      "href": "http://localhost:8080/items?page=5&size=5"
    },
  },
  "page": {
    "size": 5,
    "totalElements": 27,
    "totalPages": 6,
    "number": 2
  }
}

```

Each paged response will return links to the first, previous, next, and last page of results based on the current page using the IANA defined link relations `first`, `prev`, `next`, and `last`. If you are currently at the first page of results, no `prev` link will be rendered. The same is true for the last page of results: no `next` link will be rendered.

The paginated results also have extra data about the page settings , including the size of a page, total elements, total pages, and the page number you are currently viewing. This extra information makes it very easy for the consumer to configure UI tools like sliders or indicators to reflect the overall position the user is in viewing the data.

## Sorting

The REST API recognizes sorting parameters. To have your results sorted on a particular property, add a `sort` URL parameter with the name of the property you want to sort the results on. You can control the direction of the sort by appending a `,` to the the property name plus either `asc` or `desc`.

The following examples will sort results by title in ascending order:

```
GET /items?sort=title,asc HTTP/1.1
Host: localhost
```

To sort the results by more than one property, keep adding as many `sort=PROPERTY` parameters as you need. They will be added in the order they appear in the query string.

## Search Results

A number of endpoints of the service return search results. They usually support Level 2 (JSON) and Level 3 response (HAL+JSON). The search results contain the collection of matched items, paging information and optionally faceted aggregation results. The following describes the response structure for each format.

### HAL+JSON Search Results

Path	Type	Description
<code>_embedded</code>	Array	The HAL <code>_embedded</code> field that contains a collection of instances
<code>embedded._collectionName[]</code>	Array	The array of items instances defined a given JSON schema. The <code>collectionName</code> may vary depending on the types of items contained in the collection.
<code>aggregations[]</code>	Array	The aggregation results as defined in the <a href="#">Aggregation JSON Schema</a> . This field is optional if no faceted search is performed on the endpoint.
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket
<code>_links</code>	Object	Links to other states
<code>page</code>	Object	The page state. See <a href="#">&lt;&lt;_paging,Paging section&gt;</a>

Path	Type	Description
<code>page.size</code>	Number	The page size
<code>page.totalElements</code>	Number	The total elements matched by the search request
<code>page.totalPages</code>	Number	The total number of pages in the results
<code>page.number</code>	Number	The current page number (starts at 0)
<code>_links[]</code>	Array	Array of hypermedia links to other reachable states.

## JSON Search results

When the JSON or JSON-LD response is retrieved, the matched items are placed in an array referred by the *results* field. Aggregations results are present only faceted search is performed and paging information are returned.

Path	Type	Description
<code>results[]</code>	Array	The array of vocabulary instances that matches the search criteria
<code>aggregations[]</code>	Array	The aggregation results as defined in the <a href="#">Aggregation JSON Schema</a> . This field is optional is no faceted search is performed on the endpoint.
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket
<code>page</code>	Object	The page state. See <<_paging,Paging section>
<code>page.size</code>	Number	The page size

Path	Type	Description
<code>page.totalElements</code>	Number	The total elements matched by the search request
<code>page.totalPages</code>	Number	The total number of pages in the results
<code>page.number</code>	Number	The current page number (starts at 0)

## Aggregation JSON Schema

In many use cases, it is useful to aggregate the search results according some facets values (number of items per topic, per publisher, etc). Each facet is composed of a name, global metrics (right now only total count is supported), a set of buckets containing a label (unique value of the facet) and the total count for each label within the context of the facet and search results. The JSON schema of the Aggregation results is defined in the following table.

Path	Type	Description
<code>aggregations[]</code>	Array	The aggregation results
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket

## Resources Summary

The following resources are the core resources supported by the Semantic Registry Service. Details of these resources are described in the [Semantic Registry Resources](#) section.

Resources	Description	Operations
Capabilities	This resource describes the capabilities of the service	GET
Registers	Represents a collection of <a href="#">registers</a> which can be searched through based on search criteria. Also supports creation of a register instance.	GET, POST
Register	This resource supports retrieval update, and deletion of a <a href="#">register</a> instance	GET, PUT, DELETE

Resources	Description	Operations
Items	Represents a collection of <a href="#">register items</a> which can be searched through based on search criteria. Also supports creation of an item instance.	GET, POST
Item	This resource supports retrieval update, and deletion of an <a href="#">item</a> instance	GET, PUT, DELETE
JSON-LD Context	This resource returns the JSON-LD context associated with the JSON representation returned by the service	GET
SPARQL	This resource provides a SPARQL service query endpoint	GET, POST

The following resources are the core resources supported by the Harvester Service. Details of these resources are described in the [Harvester Service Resources](#) section.

Resources	Description	Operations
Harvester Types	This resource represents the collection of <a href="#">harvester types</a> supported by the service. An harvester type provides the parameter definitions needed to configure a harvester source	GET
Harvester Type	This resource represents a <a href="#">harvester type</a> definition, from which Harvester Source instance can be created.	GET
Harvester Sources	This resource represents the collection of <a href="#">harvester sources</a> that describe the source to be harvested. A harvester source defines the parameter values needed to configure the harvester source	GET, POST
Harvester Source	This resource represents an instance of a <a href="#">harvester source</a>	GET, PUT, DELETE

## Link Relation Types

The following table describes the list of link relation types used by the Semantic Registry Service when using a Hypermedia format such as HAL. We use the OGC namespace to define the relationships that can be reused by other services such as apiDocumentation, jsonldContext, and sparql endpoints. The links specific to the semantic registry service have the following relation type URI template: <http://www.opengis.net/rels/registry/{rel}>.

Relation type	URI	Description
<a href="#">registry:capabilities</a>	<a href="http://www.opengis.net/rels/capabilities">http://www.opengis.net/rels/capabilities</a>	Reference to the capabilities of the service
<a href="#">registry:registers</a>	<a href="http://www.opengis.net/rels/registry/registers">http://www.opengis.net/rels/registry/registers</a>	Reference to the registers collection
<a href="#">registry:register</a>	<a href="http://www.opengis.net/rels/registry/register">http://www.opengis.net/rels/registry/register</a>	Reference to a register

Relation type	URI	Description
<code>registry:items</code>	<a href="http://www.opengis.net/rels/registry/registers">http://www.opengis.net/rels/registry/registers</a>	Reference to an item collection
<code>registry:item</code>	<a href="http://www.opengis.net/rels/registry/register">http://www.opengis.net/rels/registry/register</a>	Reference to an item
<code>register:harvester</code>	<a href="http://www.opengis.net/rels/registry/harvester">http://www.opengis.net/rels/registry/harvester</a>	Reference to the harvester service
<code>ogc:sparql</code>	<a href="http://www.opengis.net/rels/sparql">http://www.opengis.net/rels/sparql</a>	Reference to a SPARQL Service Endpoint **
<code>ogc:jsonldContext</code>	<a href="http://www.opengis.net/rels/jsonldContext">http://www.opengis.net/rels/jsonldContext</a>	Reference to the JSON-LD context, that when applied to the JSON Content, transforms it to Linked Data
<code>ogc:apiDocumentation</code>	<a href="http://www.opengis.net/rels/apiDocumentation">http://www.opengis.net/rels/apiDocumentation</a>	Reference to the REST documentation of the service

The following table describes the list of link relation types used by the Harvester Service used to populate the registry. The links specific to the harvester service have the following relation type URI template: <http://www.opengis.net/rels/harvester/{rel}>.

Relation type	URI	Description
<code>harvester:service</code>	<a href="http://www.opengis.net/rels/harvester/harvester">http://www.opengis.net/rels/harvester/harvester</a>	Reference to the harvester service
<code>harvester:types</code>	<a href="http://www.opengis.net/rels/harvester/types">http://www.opengis.net/rels/harvester/types</a>	Reference to the harvester type collection
<code>harvester:type</code>	<a href="http://www.opengis.net/rels/harvester/types">http://www.opengis.net/rels/harvester/types</a>	Reference to the harvester type instance associated with a harvester source
<code>harvester:harvest</code>	<a href="http://www.opengis.net/rels/harvester/types">http://www.opengis.net/rels/harvester/types</a>	Reference to the harvest operation associated with a harvester source
<code>harvester:sources</code>	<a href="http://www.opengis.net/rels/harvester/sources">http://www.opengis.net/rels/harvester/sources</a>	Reference to the harvester sources collection

Relation type	URI	Description
harvester:results	<a href="http://www.opengis.net/rels/harvester/results">http://www.opengis.net/rels/harvester/results</a>	Reference to the harvester results collection (future extension)
harvester:objects	<a href="http://www.opengis.net/rels/harvester/results">http://www.opengis.net/rels/harvester/results</a>	Reference to the harvester objects collection (future extension)

## Level 2 REST Endpoints

The following table describes the endpoint URL paths for each resource. These paths are considered **NON-NORMATIVE** but rather **INFORMATIVE**, as changes of path templates may occur in the future, requiring updates of clients and version control of APIs. The use of hypermedia REST API is advised to be isolated from these changes. However we provide this information to support frameworks that work with Level 2 REST API (such as AngularJS)

Path	HTTP Methods	Consume	Produce
/	GET,HEAD		application/hal+json
/capabilities	GET,HEAD		* application/hal+json * application/json
/harvesters	GET,HEAD		application/hal+json
/harvesters/harvest	POST	application/json	* application/json * application/hal+json
/harvesters/sources	GET,HEAD		* application/hal+json * application/json
/harvesters/sources	POST	application/json	* application/hal+json * application/json
/harvesters/sources/{id}	DELETE		
/harvesters/sources/{id}	GET,HEAD		* application/hal+json * application/json

Path	HTTP Methods	Consume	Produce
/harvesters/sources/{id}	PUT	application/json	
/harvesters/sources/{id}/harvest	GET,HEAD		* application/hal+json * application/json
/harvesters/types	GET,HEAD		* application/hal+json * application/json
/harvesters/types/{id}	GET,HEAD		* application/hal+json * application/json
/items	GET,HEAD		* application/hal+json * application/json
/items	POST		* application/hal+json * application/json
/items/instance	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/items/{id}	DELETE		application/hal+json
/items/{id}	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/items/{id}	PUT	application/json	* application/hal+json * application/json



Path	HTTP Methods	Consume	Produce
/registers	GET,HEAD		* application/hal+json * application/json
/registers	POST	application/json	* application/hal+json * application/json
/registers/{id}	DELETE		
/registers/{id}	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/registers/{id}	PUT	application/json	* application/hal+json * application/json
/registers/{id}/items	GET,HEAD		* application/hal+json * application/json
/registers/{id}/items	POST	application/json	
/registers/{id}/sparql	GET,HEAD		* application/sparql-results+xml * application/sparql-results+json * text/csv * text/tab-separated-values * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples

Path	HTTP Methods	Consume	Produce
/registers/{id}/sparql	POST	* application/x-www-form-urlencoded * application/sparql-query	* application/sparql-results+xml * application/sparql-results+json * text/csv * text/tab-separated-values * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/registers/{registerId}/items/{itemId}	DELETE		
/registers/{registerId}/items/{itemId}	PUT		
/sparql	GET,HEAD		* application/sparql-results+xml * application/sparql-results+json * text/csv * text/tab-separated-values * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/sparql	POST	* application/x-www-form-urlencoded * application/sparql-query	* application/sparql-results+xml * application/sparql-results+json * text/csv * text/tab-separated-values * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples

## Content Negotiation

Many of the resources can serve multiple representations including:

Format	Mime type	Description
HAL+JSON	<code>application/hal+json</code>	It is the <b>default format</b> of the service. The JSON payload is compatible with JSON-LD and is aligned with the SRIM Core Ontology.
JSON-LD	<code>application/ld+json</code>	Compliant with the SRIM Core Ontology using the JSON-LD context.
RDF/XML	<code>application/rdf+xml</code>	Compliant with the SRIM Core Ontology
Turtle	<code>text/turtle</code>	Compliant with the Core SRIM Application Profile Ontology
N-Triples	<code>text/ntriples</code>	Compliant with the Core SRIM Application Profile Ontology

## Semantic Registry Resources

This section describes the list of resources made accessible by the Semantic Registry Service. The RESTful API has an entry point (service root) which provides links that represent the current state transitions supported by the service.

### Service Root

The root is the entry point of the RESTful Semantic Registry service. When the client consumes the API for the first time, it comes in contact with the root. If the HATEOAS constraint is to be considered and implemented throughout by clients, then this is the place to start. The root endpoint provides **a set of links** with well-defined relation types that correspond to the supported capabilities of the service. As the Semantic Registry Service API evolves in the future, there may be additional links, each with its own semantics defined by the type of [link relation](#).

The API is considered RESTful as it is fully discoverable **from the root** with **no prior knowledge** – meaning the client should be able to navigate the API by doing a GET on the root. Moving forward, all state changes are driven by the client using the available and discoverable transitions that the REST API provides in representations (hence Representational State Transfer).

### Accessing the root endpoint

To get access to the root of the service and get the list of links, a **GET** request is sent to the base url of the service.

### Request Structure

This request sends a 'GET' request to the base url of the service. Here the HTTP request returns the list of links supported by the service:

```
GET / HTTP/1.1
Host: localhost
```

## Query Parameters

None

## Response Structure

The request returns a HAL response (application/hal+json) with a **\_links** object that contains links associated with relation types (used as property names in JSON).

Path	Type	Description	Card.
<code>type</code>	String	ItemClass of the registry (always <code>srin:Service</code> )	1
<code>title</code>	String	Human readable title of the registry service	1
<code>description</code>	String	Human readable description of the registry service	1
<code>version</code>	String	Current version of the registry service	1
<code>category</code>	String	URL of the category of the registry service. Set right now to fixed value: <a href="http://www.opengis.net/specs/testbed12/semanticRegistry">http://www.opengis.net/specs/testbed12/semanticRegistry</a> .	1
<code>publisher[]</code>	Array	Information about the publisher(s) of this software	0..1
<code>publisher[].uri</code>	uri	URI identifier for the publisher	0..1
<code>publisher[].name</code>	string	Name of the publisher	1
<code>publisher[].type</code>	string	Type of the publisher (org:Organization or foaf:Person)	1

## Links

The following table describes the current link relation types supported by the service when using Hypermedia REST format (such as HAL).

Relation	Description
<code>registry:capabilities</code>	Refers to the capabilities of the service
<code>registry:items</code>	Refers to the item search endpoint

Relation	Description
<code>registry:sparql</code>	Refers to the SPARQL search endpoint
<code>registry:jsonldContext</code>	Refers to the JSON context of the items in the registry
<code>registry:registers</code>	Refers to the register search endpoint
<code>registry:harvester</code>	Refers to the harvesters endpoint
<code>self</code>	Refers to this endpoint.
<code>curies</code>	Refers to the curies defined for the links

### Example Response

The following example response in HAL+HJSON shows the response of the get operation at the service entry point.

HTTP/1.1 200 OK  
Content-Type: application/hal+json  
Content-Length: 1148

```
{
  "type" : "srim:Service",
  "title" : "Semantic Registry Service",
  "description" : "Semantic Registry Service prototype for OGC Testbed12",
  "category" : [ "http://www.opengis.net/specs/testbed12/semanticRegistry" ],
  "publisher" : [ {
    "name" : "Image Matters LLC",
    "uri" : "http://www.imagemattersllc.com",
    "type" : "org:Organization"
  } ],
  "version" : "0.1",
  "_links" : {
    "registry:capabilities" : {
      "href" : "http://localhost:8080/capabilities"
    },
    "registry:items" : {
      "href" : "http://localhost:8080/items"
    },
    "registry:sparql" : {
      "href" : "http://localhost:8080/sparql"
    },
    "registry:jsonldContext" : {
      "href" : "http://localhost:8080/context"
    },
    "registry:registers" : {
      "href" : "http://localhost:8080/registers"
    },
    "registry:harvester" : {
      "href" : "http://localhost:8080/harvesters"
    },
    "self" : {
      "href" : "http://localhost:8080"
    },
    "curies" : [ {
      "href" : "http://www.opengis.net/rels/registry/{rel}",
      "name" : "registry",
      "templated" : true
    } ]
  }
}
```

## Capabilities

This resource describes the capabilities of the service, including the supported application profiles, formats, and languages.

## Query Parameters

None.

## Example Request

The following is an example of a GET Request performed on the Capability resource.

```
GET /capabilities HTTP/1.1
Host: localhost
```

## Response Structure

The response of a Capability request has the following structure:

Path	Type	Description
<code>itemClasses[]</code>	Array	ItemClasses supported by the registry
<code>itemClasses[].id</code>	String	The ID of the ItemClass
<code>itemClasses[].uri</code>	String	The URI of the ItemClass
<code>itemClasses[].type</code>	String	The RDF type of the ItemClass (always <code>srin:ItemClass</code> )
<code>itemClasses[].title</code>	String	The human readable title of the ItemClass
<code>itemClasses[].description</code>	String	The human readable description of the ItemClass
<code>applicationProfiles[]</code>	Array	Application profiles supported by the registry
<code>applicationProfiles[].uri</code>	String	The URI of the application profile
<code>applicationProfiles[].label</code>	String	The human readable label of the application profile
<code>applicationProfiles[].description</code>	String	The human readable description of the application profile
<code>languages[]</code>	Array	The supported languages ( <code>dct:LinguisticSystem</code> ) supported by the vocabulary manager
<code>languages[].uri</code>	String	The uri of the language. This uri should be used as reference in the vocabulary metadata

Path	Type	Description
languages[].type	String	The RDF type of the Linguistic System (always dct:LinguisticSystem)
languages[].label	String	The human readable label of the language
languages[].iso2Code	String	The ISO 639-1 code (2 letter code)
languages[].iso3Code	String	The ISO 639-2 code (3 letter code)
_links	Object	The hypermedia links to other states

## Links

The following table describes the current link relation types in the HAL capabilities response

Relation	Description
registry:service	TODO
registry:harvesters	TODO
self	Refers to this endpoint.
curies	Refers to the curies defined for the links

## Example Response

The following is an example of the response from a GET Request performed on the Capability Resource.



```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 30456
```

```
{
  "itemClasses" : [ {
    "id" : "dcat:Dataset",
    "uri" : "http://www.w3.org/ns/dcat#Dataset",
    "type" : "srim:ItemClass",
    "title" : "Dataset",
    "description" : "Dataset class"
  }, {
    "id" : "srim:Service",
    "uri" : "http://www.opengis.net/ont/testbed12/srim#Service",
    "type" : "srim:ItemClass",
    "title" : "Service",
    "description" : "Service class"
  }, {
    ...
  }
}
```

The response is quite lengthy, so has been truncated here. The full response is available [in separate document](#).

## JSON-LD Context

The JSON produced by the Semantic Registry Service is compatible with the SRIM Application Profile by using JSON-LD context. The context is made accessible through an endpoint, so it can be referred and imported by a JSON-LD processor. The processor can then convert the context into a Linked Data representation adhering to the SRIM Profile. The JSON-LD context should be updated to accommodate the mapping of the different profiles supported by the service.

### Query Parameters

None

### Example Request

The following is an example of a GET Request performed on the JSON-LD context resource.

```
GET /context HTTP/1.1
Host: localhost
```

### Response Structure

The response of the JSON-LD Context conformst to the JSON-LD standard.

Path	Type	Description
@context	Array	Context for Item objects in the register

## Example Response

The following example response shows a JSON-LD context of the service.

```

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 10757

{
  "@context": {
    "pav": "http://purl.org/pav/",
    "dct": "http://purl.org/dc/terms/",
    "owl": "http://www.w3.org/2002/07/owl#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "srim": "http://www.opengis.net/ont/testbed/12/srim#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "dcat": "http://www.w3.org/ns/dcat#",
    "iso639-2": "http://id.loc.gov/vocabulary/iso639-2/",
    "lingvoj": "http://www.lingvoj.org/ontology#",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "ldp": "http://www.w3.org/ns/ldp#",
    "locn": "http://www.w3.org/ns/locn#",
    "adms": "http://www.w3.org/TR/vocab-adms/#",
    "extent": "http://www.opengis.net/ont/extent#",
    "idvoc": "http://www.opengis.net/ont/identifier#",
    "link": "http://www.opengis.net/ont/link#",
    "gr": "http://www.heppnetz.de/ontologies/goodrelations/v1#",
    "org": "http://www.socialml.org/ontologies/organization#",
    "schema": "http://www.opengis.net/ont/testbed12/srim/profile/schema#",
    "vcard": "http://www.w3.org/2006/vcard/ns#",
    "type": "@type",
    "uri": {
      "@id": "rdf:id",
      "@container": "@language"
    },
    "value": "@value",
    "lang": "@language",
    "results": "ldp:contains",
    "title": "dct:title",
    "titleMap": {
      "@id": "dct:title",
      "@container": "@language"
    },
    "description": "dct:description",
  }
}

```

```

"descriptionMap": {
  "@id": "dct:description",
  "@container": "@language"
},
"prefLabel": "skos:prefLabel",
"prefLabelMap": {
  "@id": "skos:prefLabel",
  "@container": "@language"
},
"notation": "skos:notation",
"notationMap": {
  "@id": "skos:notation",
  "@container": "@language"
},
"category": {
  "@id": "dct:type",
  "@type": "@id"
},
"hasVersion": "pav:hasVersion",
"hasCurrentVersion": {
  "@id": "pav:hasCurrentVersion",
  "@type": "@id"
},
"version": "pav:version",
"creator": {
  "@id": "dct:creator",
  "@type": "@id"
},
"versionNotes": "adms:versionNotes",
"versionNotesMap": {
  "@id": "adms:versionNotes",
  "@container": "@language"
},
"versionOf": {
  "@id": "pav:versionOf",
  "@type": "@id"
},
"contributor": {
  "@id": "dct:contributor",
  "@type": "@id"
},
"publisher": {
  "@id": "dct:publisher",
  "@type": "@id"
},
"language": {
  "@id": "dct:language",
  "@type": "@id"
},
"iso2Code": "lingvoj:iso1",
"iso3Code": "lingvoj:iso2",

```

```

"label": "rdfs:label",
"name": "foaf:name",
"nameMap": {
  "@id": "foaf:name",
  "@container": "@language"
},
"sameAs": {
  "@id": "owl:sameAs",
  "@type": "@id"
},
"distribution": {
  "@id": "dcat:distribution",
  "@type": "@id"
},
"license": {
  "@id": "dct:license",
  "@type": "@id"
},
"created": "dct:created",
"issued": "dct:issued",
"modified": "dct:modified",
"accessURL": {
  "@id": "dcat:accessURL",
  "@type": "@id"
},
"mediaType": "dcat:mediaType",
"country-name": "vcard:country-name",
"region": "vcard:region",
"locality": "vcard:locality",
"postal-code": "vcard:postal-code",
"street-address": "vcard:street-address",
"format": {
  "@id": "dct:format",
  "@type": "@id"
},
"agent": {
  "@id": "prov:agent",
  "@type": "@id"
},
"role": {
  "@id": "prov:role",
  "@type": "@id"
},
"inScheme": {
  "@id": "skos:inScheme",
  "@type": "@id"
},
"byteSize": "dcat:byteSize",
"conformsTo": {
  "@id": "dct:conformsTo",
  "@type": "@id"
}

```

```

},
"page": {
  "@id": "foaf:page",
  "@type": "@id"
},
"rights": {
  "@id": "dct:rights",
  "@type": "@id"
},
"representationTechnique": {
  "@id": "adms:representationTechnique",
  "@type": "@id"
},
"hasValue": {
  "@id": "vcard:hasValue",
  "@type": "@id"
},
"northBoundLatitude": "extent:northBoundLatitude",
"southBoundLatitude": "extent:southBoundLatitude",
"westBoundLongitude": "extent:westBoundLongitude",
"eastBoundLongitude": "extent:eastBoundLongitude",
"code": "id:code",
"codespace": {
  "@id": "id:codespace",
  "@type": "@id"
},
"authority": {
  "@id": "id:authority",
  "@type": "@id"
},
"landingPage": {
  "@id": "dcat:landingPage",
  "@type": "@id"
},
"accrualPeriodicity": {
  "@id": "dct:accrualPeriodicity",
  "@type": "@id"
},
"purpose": "srim:purpose",
"accessLevel": {
  "@id": "srim:accessLevel",
  "@type": "@id"
},
"itemClass": {
  "@id": "srim:itemClass",
  "@type": "@id"
},
"keyword": "dcat:keyword",
"identifier": "dct:identifier",
"hasLink": {
  "@id": "link:hasLink",

```

```

    "@type": "@id"
  },
  "theme": {
    "@id": "dcat:theme",
    "@type": "@id"
  },
  "audience": {
    "@id": "dct:audience",
    "@type": "@id"
  },
  "function": {
    "@id": "srim:function",
    "@type": "@id"
  },
  "subject": {
    "@id": "dct:subject",
    "@type": "@id"
  },
  "project": {
    "@id": "foaf:project",
    "@type": "@id"
  },
  "hasIdentifier": {
    "@id": "id:hasIdentifier",
    "@type": "@id"
  },
  "depiction": {
    "@id": "foaf:depiction",
    "@type": "@id"
  },
  "provenance": {
    "@id": "dct:provenance",
    "@type": "@id"
  },
  "contactPoint": {
    "@id": "dcat:contactPoint",
    "@type": "@id"
  },
  "hasGeographicExtent": {
    "@id": "extent:hasGeographicExtent",
    "@type": "@id"
  },
  "temporal": {
    "@id": "dct:temporal",
    "@type": "@id"
  },
  "spatial": {
    "@id": "dct:spatial",
    "@type": "@id"
  },
  "accessRights": {

```

```

    "@id": "dct:accessrights",
    "@type": "@id"
  },
  "qualifiedAttribution": {
    "@id": "prov:qualifiedAttribution",
    "@type": "@id"
  },
  "rightsHolder": {
    "@id": "dct:rightsHolder",
    "@type": "@id"
  },
  "servicedBy": {
    "@id": "srim:servicedBy",
    "@type": "@id"
  },
  "operatesOn": {
    "@id": "srim:operatesOn",
    "@type": "@id"
  },
  "relation": {
    "@id": "dct:relation",
    "@type": "@id"
  },
  "technicalStandard": {
    "@id": "srim:technicalStandard",
    "@type": "@id"
  },
  "hrefLang": {
    "@id": "link:hrefLang",
    "@type": "@id"
  },
  "href": {
    "@id": "link:href",
    "@type": "@id"
  },
  "rel": {
    "@id": "link:rel",
    "@type": "@id"
  },
  "geographicName": "locn:geographicName",
  "opens": "gr:opens",
  "closes": "gr:closes",
  "hasOpeningHoursDayOfWeek": "gr:hasOpeningHoursDayOfWeek",
  "subOrganizationOf": {
    "@id": "org:subOrganizationOf",
    "@type": "@id"
  },
  "startDate": "schema:startDate",
  "endDate": "schema:endDate",
  "alias": "srim:alias",
  "themeTaxonomy": {

```

```

    "@id": "dcat:themeTaxonomy",
    "@type": "@id"
  },
  "entry": {
    "@id": "srim:entry",
    "@type": "@id"
  },
  "item": {
    "@id": "srim:item",
    "@type": "@id"
  },
  "status": "srim:status",
  "previousVersion": {
    "@id": "pav:previousVersion",
    "@type": "@id"
  },
  "wasGeneratedBy": {
    "@id": "pav:wasGeneratedBy",
    "@type": "@id"
  },
  "source": {
    "@id": "schema:source",
    "@type": "@id"
  },
  "primaryTopic": {
    "@id": "foaf:primaryTopic",
    "@type": "@id"
  },
  "itemType": {
    "@id": "srim:itemType",
    "@type": "@id"
  },
  "organization-name": "vcard:organization-name",
  "address": {
    "@id": "vcard:address",
    "@type": "@id"
  },
  "fn": "vcard:fn",
  "hasEmail": {
    "@id": "vcard:hasEmail",
    "@type": "@id"
  },
  "vcardTitle": "vcard:title",
  "hasTelephone": {
    "@id": "vcard:hasTelephone",
    "@type": "@id"
  },
  "hasOpeningHoursSpecification": {
    "@id": "gr:hasOpeningHoursSpecification",
    "@type": "@id"
  },

```



```

    "target": {
      "@id": "schema:target",
      "@type": "@id"
    },
    "mappingLanguage": {
      "@id": "schema:mappingLanguage",
      "@type": "@id"
    },
    "namespace": {
      "@id": "schema:namespace",
      "@type": "@id"
    },
    "alternative": "dct:alternative",
    "releaseIssued": "dct:issued",
    "characterSet": {
      "@id": "srim:characterSet",
      "@type": "@id"
    },
    "source": {
      "@id": "dct:source",
      "@type": "@id"
    }
  }
}

```

## Registers

A **Register** is a curated collection of metadata about items stored in a repository (registry) to enable search and discovery of items. The **Registers Resource** represents a collection of Register instances, and it is used to create and list registers. For example, one register can contain datasets with associated services, while another manages schema and schema mappings, and another manages portrayal information.

### Register JSON Schema

The JSON representation of the Register instance is defined by the following JSON schema that are accessible at the following endpoints:

Path	Type	Description
<code>id</code>	<code>String</code>	Internal identifier for the register, used by the registry
<code>uri</code>	<code>String</code>	Unique identifier for the register
<code>alias</code>	<code>String</code>	Alias (alternate name) for the register
<code>type</code>	<code>String</code>	The RDF type of the register

<b>Path</b>	<b>Type</b>	<b>Description</b>
title	String	The title of the register
titleMap	Object	The title map for each language of the title
description	String	The description of the register
descriptionMap	Object	The description map for each language of the description
created	String	The date of creation of the register in the service (XSD datetime format)
modified	String	The date of the last modification of the register in the service (XSD datetime format)
issued	String	The date of publication of the register (XSD datetime format)
landingPage	String	URL pointing to the landing page of the register
language	Array	Languages of the register
themeTaxonomy	Array	Theme taxonomy of the register (array of ConceptSchemes)
registerEntry	Array	The non-Item entries in the register
itemClass	Array	The ItemClasses of the items in the register
item	Array	The items in the register
publisher	Array	The agent that published the register
accessRights	Object	The rights statement associated with the register
temporal	Array	Period of time relevant to the register
spatial	Array	Locations relevant to the register
conformsTo	Object	Standard the register conforms to
license	Object	License document associated with the register

Path	Type	Description
<code>_links</code>	Object	The hypermedia links to other states

## Search registers

A **GET** request on the Registers resource provides a list of all of the registers currently managed by the service.

### Query Parameters

Parameter	Description
<code>q</code>	Text to search in textual fields
<code>category</code>	One or more categories
<code>lang</code>	One or more languages
<code>uri</code>	One or more URIs of register instances
<code>id</code>	One or more id of register instances
<code>includeFacet</code>	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.
<code>facet.fieldname</code>	Constraint values of a given facet field name
<code>constraint</code>	A constraint expressed in CQL. This is used to express more advanced query filtering
<code>fields</code>	One or more fields to be included in the response. Use JSON path dot notation for reference paths (ex: publisher.name)
<code>pageNumber</code>	The number of the current page as defined in the <a href="#">Paging section</a>
<code>pageSize</code>	The count of items on the current page as defined in the <a href="#">Paging section</a>
<code>sort</code>	The sorting parameters as defined in the <a href="#">Sorting section</a> .

### Example request

The following is an example of a GET Request performed on the Registers resource requesting the first page composed of 3 items, which is sorted by title in ascending order.

```
GET /registers?sort=title%2Case HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response is structured according to the [Register JSON Schema](#). The default response returns a summary of the register with references to id and title.

### Links

The following link relation types are provided in the response to allow the transition to other states from the register collection embedded in the response.

Relation	Description
<a href="#">self</a>	Refers to the registers list itself
<a href="#">registry:service</a>	Refers to the root of the registry
<a href="#">curies</a>	Refers to the curies defined for the links
<a href="#">first</a>	The first page of results
<a href="#">last</a>	The last page of results
<a href="#">next</a>	The next page of results
<a href="#">prev</a>	The previous page of results

### Example response

=====  
HAL Representation

The embedded objects in the response of the request conform to the Register Model defined in the [Registry JSON Schema](#). The rest of the response return paging information and links to other states. The HAL defined container for the collection of results is called **registry:registers** and is placed under the HAL **\_embedded** property.

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 502
{
  "_embedded": {
    "registry:registers": [{
      "id": "portrayal",
      "type": "srim:Register",
      "title": "Portrayal Register",
      "description": "OGC Testbed 12 Register for portrayal information",
      "_links": {
        "self": {
          "href": "http://localhost:8080/registers/portrayal"
        }
      }
    }
  ]
}
```

```

    }
  }, {
    "id": "schemas",
    "type": "srim:Register",
    "title": "Schema Register",
    "description": "Register for schemas",
    "_links": {
      "self": {
        "href": "http://localhost:8080/registers/schemas"
      }
    }
  }, {
    "id": "datasets",
    "type": "srim:Register",
    "title": "Testbed12 Datasets Register",
    "description": "Testbed 12 catalog for datasets and services harvested
from multiple CSW instances",
    "_links": {
      "self": {
        "href": "http://localhost:8080/registers/datasets"
      }
    }
  ]
},
"_links": {
  "self": {
    "href": "http://localhost:8080/registers?sort=title,asc"
  },
  "registry:service": {
    "href": "http://localhost:8080"
  },
  "curies": [{
    "href": "http://www.opengis.net/rels/registry/{rel}",
    "name": "registry",
    "templated": true
  }]
},
"aggregations": [],
"page": {
  "size": 20,
  "totalElements": 3,
  "totalPages": 1,
  "number": 0
}
}

```

===== JSON Representation

The JSON Representation defines the same content as HAL without the links and the `_embedded` tag. The content is placed in the results array.

```

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 352
{
  "results": [{
    "id": "portrayal",
    "type": "srim:Register",
    "title": "Portrayal Register",
    "description": "OGC Testbed 12 Register for portrayal information"
  }, {
    "id": "schemas",
    "type": "srim:Register",
    "title": "Schema Register",
    "description": "Register for schemas"
  }, {
    "id": "datasets",
    "type": "srim:Register",
    "title": "Testbed12 Datasets Register",
    "description": "Testbed 12 catalog for datasets and services harvested from
multiple CSW instances"
  }],
  "page": {
    "size": 20,
    "number": 0,
    "totalElements": 3
  }
}

```

## Creating a Register

A **POST** request is used to create a new Register. The response from this request should have a status code of **201 Created** and contain a **Location** header whose value is the URI of the newly created register. To access the register's details, either perform a **GET** Request on the URI in the **Location** header of the response, or just access the details of the register in the body of the response. We decided to have the API return the created representation in the response to prevent an API consumer from having to hit the API again for the newly created representation.

### Request structure

The JSON body of the POST request adheres to the [Register JSON Schema](#). The fields are detailed in the following table:

Path	Type	Description
<b>id</b>	<b>String</b>	Internal identifier for the register, used by the registry
<b>uri</b>	<b>String</b>	Unique identifier for the register

<b>Path</b>	<b>Type</b>	<b>Description</b>
alias	String	Alias (alternate name) for the register
type	String	The RDF type of the register
title	String	The title of the register
titleMap	Object	The title map for each language of the title
description	String	The description of the register
descriptionMap	Object	The description map for each language of the description
created	String	The date of creation of the register in the service (XSD datetime format)
modified	String	The date of the last modification of the register in the service (XSD datetime format)
issued	String	The date of publication of the register (XSD datetime format)
landingPage	String	URL pointing to the landing page of the register
language	Array	Languages of the register
themeTaxonomy	Array	Theme taxonomy of the register (array of ConceptSchemes)
registerEntry	Array	The non-Item entries in the register
itemClass	Array	The ItemClasses of the items in the register
item	Array	The items in the register
publisher	Array	The agent that published the register
accessRights	Object	The rights statement associated with the register
temporal	Array	Period of time relevant to the register
spatial	Array	Locations relevant to the register
conformsTo	Object	Standard the register conforms to

Path	Type	Description
<a href="#">license</a>	Object	License document associated with the register

### Example request

The following POST request creates a new Register with the title 'NGDA Datasets Register`

```
POST /registers HTTP/1.1
Content-Type: application/json
Accept: application/hal+json
Host: localhost
Content-Length: 202

{
  "id" : "urn:example-register-44",
  "uri" : "urn:example-register-44",
  "type" : "srim:Register",
  "title" : "Vocabulary Register",
  "description" : "Register for Controlled Vocabularies"
}
```

### Example response

The response returns the URI of the newly created Register in the [Location](#) header with a '201 Created' status code as well as the created Register details in the response body.



```
HTTP/1.1 201 Created
Location: http://localhost:8080/registers/urn:example-register-44
Content-Type: application/hal+json
Content-Length: 895
```

```
{
  "id" : "urn:example-register-44",
  "uri" : "urn:example-register-44",
  "type" : "srim:Register",
  "title" : "Vocabulary Register",
  "description" : "Register for Controlled Vocabularies",
  "issued" : "2016-10-17",
  "created" : "2016-10-17T19:28:38.664Z",
  "modified" : "2016-10-17T19:28:38.664Z",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/registers"
    },
    "registry:registers" : {
      "href" : "http://localhost:8080/registers"
    },
    "registry:registry" : {
      "href" : "http://localhost:8080"
    },
    "registry:sparql" : {
      "href" : "http://localhost:8080/sparql"
    },
    "registry:items" : {
      "href" : "http://localhost:8080/registers/items"
    },
    "curies" : [ {
      "href" : "http://www.opengis.net/rels/registry/{rel}",
      "name" : "registry",
      "templated" : true
    } ]
  }
}
```

## Register

The Register resource is used to retrieve, update, and delete a register instance. A **Register** is defined as a curated collection of metadata about items stored in a repository (registry) to enable search and discovery of resources. This class is used as a container to exchange registers of resources between registries. Registers can manage different types of resource items such as datasets, map layers or services.

### Retrieve a register

A HTTP **GET** Request will retrieve a particular instance of register.

## Query Parameters

None

## Example request

The following HTTP request performs a GET Request to get an instance of a register:

```
GET /registers/schemas HTTP/1.1
Accept: application/hal+json
Host: localhost
```

## Response structure

The response is structured according to the [Register JSON Schema](#).

## Links

The following table shows the link relation types accessible from a register instance that provide transitions to other states related to the register instance.

Relation	Description
<a href="#">self</a>	Refers to this <a href="#">item</a> itself
<a href="#">registry:registers</a>	Refers to registers search endpoint
<a href="#">registry:registry</a>	Refers to the root of the registry
<a href="#">registry:items</a>	Refers to the items within the registry
<a href="#">registry:sparql</a>	Refers to the SPARQL endpoint
<a href="#">curies</a>	Refers to the curies defined for the links

## Example responses

===== HAL Representation The response of the request conforms to the [Register JSON Schema](#) with additional links.

HTTP/1.1 200 OK  
Content-Type: application/hal+json  
Content-Length: 895

```
{
  "id": "schemas",
  "type": "srim:Register",
  "title": "Schema Register",
  "description": "Register for schemas",
  "issued": "2016-11-08",
  "created": "2016-11-08T02:51:42.981Z",
  "modified": "2016-11-08T02:51:42.981Z",
  "publisher": [{
    "name": "Image Matters LLC",
    "uri": "http://www.imagemattersllc.com",
    "type": "org:Organization"
  }],
  "itemClass": [
    "schema:SchemaMapping",
    "schema:Schema"
  ],
  "_links": {
    "self": {
      "href": "http://localhost:8080/registers/schemas"
    },
    "registry:registers": {
      "href": "http://localhost:8080/registers"
    },
    "registry:registry": {
      "href": "http://localhost:8080"
    },
    "registry:sparql": {
      "href": "http://localhost:8080/schemas/sparql"
    },
    "registry:items": {
      "href": "http://localhost:8080/registers/schemas/items"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/registry/{rel}",
      "name": "registry",
      "templated": true
    }]
  }
}
```

===== JSON Representation

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 523
```

```
{
  "id": "schemas",
  "type": "srim:Register",
  "title": "Schema Register",
  "description": "Register for schemas",
  "issued": "2016-11-08",
  "created": "2016-11-08T02:51:42.981Z",
  "modified": "2016-11-08T02:51:42.981Z",
  "publisher": [{
    "name": "Image Matters LLC",
    "uri": "http://www.imagemattersllc.com",
    "type": "org:Organization"
  }],
  "itemClass": [
    "schema:SchemaMapping",
    "schema:Schema"
  ]
}
```

#### ===== Turtle Representation

```
HTTP/1.1 200 OK
Content-Type: text/turtle
Content-Length: 1366
```

```
@prefix org: <http://www.socialml.org/ontologies/organization#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix srim: <http://www.opengis.net/ont/testbed12/srim#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .
```

```
[ a srim:Register ;
  dct:created "2016-10-10T16:08:19.979Z"^^xsd:dateTime ;
  dct:description "Register for schemas" ;
  dct:issued "2016-10-10T16:08:19.979Z"^^xsd:date ;
  dct:modified "2016-10-10T16:08:19.979Z"^^xsd:dateTime ;
  dct:publisher <http://www.imagemattersllc.com> ;
  dct:title "Schema Register" ;
  srim:itemClass <schema:SchemaMapping> , <schema:Schema>
] .
```

```
<http://www.imagemattersllc.com>
  a org:Organization ;
  foaf:name "Image Matters LLC" .
```

## ===== RDF/XML Representation

```
HTTP/1.1 200 OK
Content-Type: application/rdf+xml
Content-Length: 1609

<?xml version='1.0'?>
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:org='http://www.socialml.org/ontologies/organization#'
  xmlns:dct='http://purl.org/dc/terms/'
  xmlns:srim='http://www.opengis.net/ont/testbed12/srim#'
  xmlns:schema='http://www.opengis.net/ont/testbed12/srim/profile/schema#'
  xmlns:dcat='http://www.w3.org/ns/dcat#'
  xmlns:foaf='http://xmlns.com/foaf/0.1/'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema#'>
  <srim:Register>
    <dct:title>Schema Register</dct:title>
    <dct:description>Register for schemas</dct:description>
    <dct:issued rdf:datatype='http://www.w3.org/2001/XMLSchema#date'
    >2016-11-07T21:51:42.981-05:00</dct:issued>
    <dct:modified rdf:datatype='http://www.w3.org/2001/XMLSchema#dateTime'
    >2016-11-07T21:51:42.981-05:00</dct:modified>
    <dct:created rdf:datatype='http://www.w3.org/2001/XMLSchema#dateTime'
    >2016-11-07T21:51:42.981-05:00</dct:created>
    <srim:itemClass rdf:resource='schema:SchemaMapping' />
    <srim:itemClass rdf:resource='schema:Schema' />
    <dct:publisher>
      <org:Organization rdf:about='http://www.imagemattersllc.com'>
        <foaf:name>Image Matters LLC</foaf:name>
      </org:Organization>
    </dct:publisher>
  </srim:Register>
</rdf:RDF>
```

### Update a Register

A HTTP **PUT** request, with a body containing a JSON representation of the Register instance that adheres to the [Register JSON Schema](#), is used to update a Register instance. The response of the request returns the HTTP Status Code 202 (accepted) if the update is accepted.

### Request Structure

The request body of the register instance is defined by the [Register JSON Schema](#).

### Example Request

The following HTTP request performs a register update using the JSON representation of the register:

```
PUT /registers/urn:example-register-18 HTTP/1.1
Content-Type: application/json
Accept: application/hal+json
Host: localhost
Content-Length: 315

{
  "id" : "urn:example-register-18",
  "uri" : "urn:example-register-18",
  "type" : "srim:Register",
  "title" : "New updated title",
  "description" : "Register for Controlled Vocabularies",
  "issued" : "2016-10-17",
  "created" : "2016-10-17T19:28:16.168Z",
  "modified" : "2016-10-17T19:28:16.168Z"
}
```

### Example Response

===== HAL+JSON Representation

The response of the request returns the HTTP Status Code 202 (accepted) if the update is accepted, as well as the updated item object so the client can avoid making a second call to get the register. The server should add or update the *modified* field.

```
HTTP/1.1 202 Accepted
Content-Type: application/hal+json
Content-Length: 893
```

```
{
  "id" : "urn:example-register-18",
  "uri" : "urn:example-register-18",
  "type" : "srim:Register",
  "title" : "New updated title",
  "description" : "Register for Controlled Vocabularies",
  "issued" : "2016-10-17",
  "created" : "2016-10-17T19:28:16.168Z",
  "modified" : "2016-10-17T19:28:16.181Z",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/registers"
    },
    "registry:registers" : {
      "href" : "http://localhost:8080/registers"
    },
    "registry:registry" : {
      "href" : "http://localhost:8080"
    },
    "registry:sparql" : {
      "href" : "http://localhost:8080/sparql"
    },
    "registry:items" : {
      "href" : "http://localhost:8080/registers/items"
    },
    "curies" : [ {
      "href" : "http://www.opengis.net/rels/registry/{rel}",
      "name" : "registry",
      "templated" : true
    } ]
  }
}
```

#### ==== JSON Representation

The JSON response of the request returns the HTTP Status Code 202 (accepted) if the update is accepted, and the updated item object without the hypermedia links. The server should add or update the *modified* field.

```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 315
```

```
{
  "id" : "urn:example-register-36",
  "uri" : "urn:example-register-36",
  "type" : "srim:Register",
  "title" : "New updated title",
  "description" : "Register for Controlled Vocabularies",
  "issued" : "2016-10-17",
  "created" : "2016-10-17T19:28:33.740Z",
  "modified" : "2016-10-17T19:28:33.759Z"
}
```

## Delete a Register

A **DELETE** request is used to delete a register. The deletion will remove all the dependent resources from the registry (entries and resource items, harvesters). If the delete operation is successful, the response returns the HTTP code 200 and if the delete operation is unsuccessful, the response returns the HTTP code 404 (Not Found). This functionality is available only for authorized users.

### Example request

The following HTTP request performs a HTTP Delete request on a register instance URL.

```
DELETE /registers/urn:example-register-12 HTTP/1.1
Host: localhost
```

### Example response

If the register is successfully deleted the response will return the code 200 shown below.

```
HTTP/1.1 200 OK
```

## Items

The **Items Resource** is used to create and list items. An **Item** is the top class of the SRIM hierarchy. It is a subclass of the class `rdf:Resource`. It represents any information entities or services that can be registered in a register. Each `srim:Item` is associated with an **ItemClass** that is supported by the service. Example of ItemClasses include `dc:Dataset`, `srim:Service`, `schema:Schema`, and `symbol:Symbol`. The ItemClasses and the application profiles supported by the service are defined in the Capabilities.

## JSON Schema

As the registry service can support different item classes and application profiles, the JSON schema



supported by the service is not unique. However there is a set of core properties that will always be present for any items in the registry. The following table describes the core minimal properties for an item.

Path	Type	Description	Cardinality
id	String	Internal identifier for the item, used by the registry	1
uri	String	Linked Data URI for the item (equivalent to @id in JSON-LD)	0..1
type	String	The Item class identifier as published in the capability.	1
title	String	The title of the item	1
titleMap	Object	The title map for each language of the title . Each key corresponds to the two letter language identifier name defined in the capabilities ( for example "en")	0..1
description	String	The description of the item	0..1
descriptionMap	Object	The description map for each language of the description. Each key corresponds to the two letter language identifier name defined in the capabilities ( for example "en")	0..1
created	String	The date of creation (XSD datetime format) (generated by the service)	1
modified	String	The date of the last modification (XSD datetime format) (generated by the service)	0..1
register	Array	Array of register identifiers in which the item belongs to.	0..1

## Search Items

There are two ways to search for Items - globally or constrained to a given Register. The global search is done by performing a HTTP **GET** Request on the Items Resource, while the register constrained search is done by performing a HTTP **GET** Request on the Register Items Resource. The item search supports free text, spatial, temporal or semantic search and can return results with aggregations on specific facets. Additionally, if the search is performed from a given register, the parameter registerId is ignored, and if the search is performed globally, the search can be performed on multiple registries by providing a list of register aliases.

### Query Parameters

The following parameters are supported in the query:

Parameter	Description
q	Text to search in textual fields
category	One or more categories
lang	One or more languages (using 2-letter code defined in the capabilities)
uri	One or more URIs of items instances
id	One or more id of item instances
type	One or more item class identifiers defining the type of items to fetch.
register	One or more id of register instances
bbox	Bounding box defined by minx,miny,maxx,maxy . This option may be deprecated in the future and replaced with a CQL constraint
within	Boolean defining if the search needs be strictly within the bbox or not. This option may be deprecated in the future and replaced with a CQL constraint
includeFacet	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, comma delimited field names can be set.
facet.fieldname	Constraint values of a given facet field name
constraint	A constraint expressed in CQL. This is used to express more advanced query filtering
fields	One or more fields to be included in the response. Use JSON path dot notation for reference paths (ex: publisher.name)

Parameter	Description
<code>pageNumber</code>	The number of the current page as defined in the <a href="#">Paging section</a>
<code>pageSize</code>	The count of items on the current page as defined in the <a href="#">Paging section</a>
<code>sort</code>	The sorting parameters as defined in the <a href="#">Sorting section</a> .

### Example requests

The following request performs a search of instance of item Class `dc:Dataset`.

```
GET /registers/urn:example-register-27/items?type=dc:Dataset HTTP/1.1
Accept: application/hal+json
Host: localhost
```

The following request performs a search of items with type `dc:Dataset` (item class id) containing the word `arctic`, includes aggregation in all facets, and returns results with a page size of 4 elements.

```
GET /items?q=Arctic&type=dc:Dataset&size=4&includeFacet=true HTTP/1.1
Host: localhost
```

### Response structure

The response is structured according to the [Item JSON Schema](#).

### Links

The following link relation types are provided in the response to allow the transition to others states from the register item embedded in the response.

Relation	Description
<code>self</code>	Refers to the items list itself
<code>registry:service</code>	Refers to the root of the registry
<code>curies</code>	Refers to the curies defined for the links
<code>first</code>	The first page of results
<code>last</code>	The last page of results
<code>next</code>	The next page of results
<code>prev</code>	The previous page of results

## Example response

### HAL+JSON Representation

The embedded objects in the response of the request conform to the [Register Item JSON Schema](#). The rest of the response returns paging and aggregation information as well as links to other states.

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 2352
{
  "_embedded": {
    "registry:items": [{
      "id": "2fe9867e7a22473bc2ec25dac3d959c7",
      "uri": "urn:x-envitia:TB12:NMF:nmf-G1470659423089",
      "type": "dcat:Dataset",
      "title": "Murre as indicators of a changing Arctic",
      "description": "The two species of murre, thick-billed Uria lomvia and
common U. aalge, both have circumpolar distributions, breeding in Arctic, sub-Arctic
and temperate seas from alifornia and N Spain to N Greenland, high Arctic Canada,
Svalbard, Franz Josef Land and Novaya Zemlya (Box 4.3 Fig. 1). Conservation of Arctic
Flora and Fauna, CAFF 2013 - Akureyri . Arctic Biodiversity Assessment. Status and
Trends in Arctic biodiversity. - Birds(Chapter 4) page 163",
      "created": "2016-10-10T16:09:47.684Z",
      "_links": {
        "self": {
          "href":
"http://54.208.90.94/registry/items/2fe9867e7a22473bc2ec25dac3d959c7"
        }
      }
    }, {
      "id": "2373c7d58a4d54f0093bfb9b9f9d6c21",
      "uri": "urn:x-envitia:TB12:NMF:nmf-G1470659422925",
      "type": "dcat:Dataset",
      "title": "Avian biodiversity in different regions of the Arctic",
      "description": "Figure 4.1. Avian biodiversity in different regions of the
Arctic. Charts on the inner circle show species numbers of different bird groups in
the high Arctic, on the outer circle in the low Arctic. The size of the charts is
scaled to the number of species in each region, which ranges from 32 (Svalbard) to 117
(low Arctic Alaska). CAFF 2013. Arctic Biodiversity Assessment. Status and Trends in
Arctic biodiversity. Conservation of Arctic Flora and Fauna, Akureyri - Birds (Chapter
4) page 145",
      "created": "2016-10-10T16:09:45.791Z",
      "_links": {
        "self": {
          "href":
"http://54.208.90.94/registry/items/2373c7d58a4d54f0093bfb9b9f9d6c21"
        }
      }
    }, {
      "id": "a267316b5c5171893b1771e71b7c5a19",
```

```

    "uri": "urn:x-envitia:TB12:NMF:nmf-G1470659423365",
    "type": "dcat:Dataset",
    "title": "Cumulative numbers of marine fish.",
    "description": "Cumulative numbers of marine fish diversity (n = 633,
Appendix 6.2) in the Arctic Ocean and adjacent seas (AOAS) from 1758 to the present.
Species are broadly grouped according to zoogeographic pattern (cf. Section 6.3.1):
Arctic (A, blue symbols) and non-Arctic (Ë AB, B, WD, red symbols). Grey bars denote
periods with many descriptions of new Arctic species. Note that 75% of the non-Arctic
species known to science were described by 1912, whereas the same proportion for
Arctic species was only reached in 1976. See text for further information.
Conservation of Arctic Flora and Fauna, CAFF 2013 - Akureyri . Arctic Biodiversity
Assessment. Status and Trends in Arctic biodiversity. - Fishes(Chapter 6) page 220",
    "created": "2016-10-10T16:09:50.824Z",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/registry/items/a267316b5c5171893b1771e71b7c5a19"
      }
    }
  }, {
    "id": "d52e1c753f61dc577dce3ea5c5b818c8",
    "uri": "urn:x-envitia:TB12:NMF:nmf-G1470659423695",
    "type": "dcat:Dataset",
    "title": "Diversity of Arctic marine phytoplankton: based on surveys in
the Russian Arctic",
    "description": "The number of species depends partly on what has been
studied. Proportions vary somewhat around the Arctic, but diatoms and dinoflagellates
are the most diverse groups everywhere. The greatest sampling effort has been in the
Laptev Sea, Hudson Bay, and the Norwegian sector of the Barents Sea. Species shown are
among the most commonly recorded. Published in the Life Linked to Ice released in
2013, page 26. Life Linked to Ice: A guide to sea-ice-associated biodiversity in this
time of rapid change. CAFF Assessment Series No. 10. Conservation of Arctic Flora and
Fauna, Iceland. ISBN: 978-9935-431-25-7.",
    "created": "2016-10-10T16:09:54.199Z",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/registry/items/d52e1c753f61dc577dce3ea5c5b818c8"
      }
    }
  }
}],
  "_links": {
    "first": {
      "href":
"http://54.208.90.94/registry/items?q=Arctic&type=dcat:Dataset&includeFacet=true&page=
0&size=4"
    },
    "self": {
      "href":
"http://54.208.90.94/registry/items?q=Arctic&type=dcat:Dataset&size=4&includeFacet=tru

```

```

e"
    },
    "next": {
      "href":
"http://54.208.90.94/registry/items?q=Arctic&type=dc:Dataset&includeFacet=true&page=
1&size=4"
    },
    "last": {
      "href":
"http://54.208.90.94/registry/items?q=Arctic&type=dc:Dataset&includeFacet=true&page=
33&size=4"
    },
    "registry:service": {
      "href": "http://54.208.90.94/registry"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/registry/{rel}",
      "name": "registry",
      "templated": true
    }
  ],
  "aggregations": [{
    "name": "subject",
    "metrics": {
      "count": 1
    },
    "buckets": [{
      "label": "DATASET",
      "count": 1
    }
  ]
}, {
  "name": "language",
  "metrics": {
    "count": 1
  },
  "buckets": [{
    "label": "iso3:eng",
    "count": 1
  }
]
}, {
  "name": "type",
  "metrics": {
    "count": 135
  },
  "buckets": [{
    "label": "dc:Dataset",
    "count": 135
  }
]
}, {
  "name": "keyword",
  "metrics": {

```

```

    "count": 1
  },
  "buckets": [{
    "label": "dataset",
    "count": 1
  }]
}, {
  "name": "register",
  "metrics": {
    "count": 135
  },
  "buckets": [{
    "label": "datasets",
    "count": 135
  }]
}],
"page": {
  "size": 4,
  "totalElements": 135,
  "totalPages": 34,
  "number": 0
}
}

```

## JSON Representation

```

HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 1920
{
  "results": [{
    "id": "2fe9867e7a22473bc2ec25dac3d959c7",
    "uri": "urn:x-envitia:TB12:NMF:nmf-G1470659423089",
    "type": "dcat:Dataset",
    "title": "Murre as indicators of a changing Arctic",
    "description": "The two species of murre, thick-billed Uria lomvia and common
    U. aalge, both have circumpolar distributions, breeding in Arctic, sub-Arctic and
    temperate seas from California and N Spain to N Greenland, high Arctic Canada,
    Svalbard, Franz Josef Land and Novaya Zemlya (Box 4.3 Fig. 1). Conservation of Arctic
    Flora and Fauna, CAFF 2013 - Akureyri . Arctic Biodiversity Assessment. Status and
    Trends in Arctic biodiversity. - Birds(Chapter 4) page 163",
    "created": "2016-10-10T16:09:47.684Z"
  }, {
    "id": "2373c7d58a4d54f0093bfb9b9f9d6c21",
    "uri": "urn:x-envitia:TB12:NMF:nmf-G1470659422925",
    "type": "dcat:Dataset",
    "title": "Avian biodiversity in different regions of the Arctic",
    "description": "Figure 4.1. Avian biodiversity in different regions of the
    Arctic. Charts on the inner circle show species numbers of different bird groups in
    the high Arctic, on the outer circle in the low Arctic. The size of the charts is

```

scaled to the number of species in each region, which ranges from 32 (Svalbard) to 117 (low Arctic Alaska). CAFF 2013. Arctic Biodiversity Assessment. Status and Trends in Arctic biodiversity. Conservation of Arctic Flora and Fauna, Akureyri - Birds (Chapter 4) page 145",

"created": "2016-10-10T16:09:45.791Z"

}, {

"id": "a267316b5c5171893b1771e71b7c5a19",

"uri": "urn:x-envitia:TB12:NMF:nmf-G1470659423365",

"type": "dcat:Dataset",

"title": "Cumulative numbers of marine fish.",

"description": "Cumulative numbers of marine fish diversity (n = 633, Appendix 6.2) in the Arctic Ocean and adjacent seas (AOAS) from 1758 to the present. Species are broadly grouped according to zoogeographic pattern (cf. Section 6.3.1): Arctic (A, blue symbols) and non-Arctic (AB, B, WD, red symbols). Grey bars denote periods with many descriptions of new Arctic species. Note that 75% of the non-Arctic species known to science were described by 1912, whereas the same proportion for Arctic species was only reached in 1976. See text for further information. Conservation of Arctic Flora and Fauna, CAFF 2013 - Akureyri . Arctic Biodiversity Assessment. Status and Trends in Arctic biodiversity. - Fishes(Chapter 6) page 220",

"created": "2016-10-10T16:09:50.824Z"

}, {

"id": "d52e1c753f61dc577dce3ea5c5b818c8",

"uri": "urn:x-envitia:TB12:NMF:nmf-G1470659423695",

"type": "dcat:Dataset",

"title": "Diversity of Arctic marine phytoplankton: based on surveys in the Russian Arctic",

"description": "The number of species depends partly on what has been studied. Proportions vary somewhat around the Arctic, but diatoms and dinoflagellates are the most diverse groups everywhere. The greatest sampling effort has been in the Laptev Sea, Hudson Bay, and the Norwegian sector of the Barents Sea. Species shown are among the most commonly recorded. Published in the Life Linked to Ice released in 2013, page 26. Life Linked to Ice: A guide to sea-ice-associated biodiversity in this time of rapid change. CAFF Assessment Series No. 10. Conservation of Arctic Flora and Fauna, Iceland. ISBN: 978-9935-431-25-7.",

"created": "2016-10-10T16:09:54.199Z"

}],

"aggregations": [{

"name": "subject",

"metrics": {

"count": 1

},

"buckets": [{

"label": "DATASET",

"count": 1

}]

}, {

"name": "language",

"metrics": {

"count": 1

},

"buckets": [{



```

        "label": "iso3:eng",
        "count": 1
    }],
    {
        "name": "type",
        "metrics": {
            "count": 135
        },
        "buckets": [{
            "label": "dcat:Dataset",
            "count": 135
        }]
    }, {
        "name": "keyword",
        "metrics": {
            "count": 1
        },
        "buckets": [{
            "label": "dataset",
            "count": 1
        }]
    }, {
        "name": "register",
        "metrics": {
            "count": 135
        },
        "buckets": [{
            "label": "datasets",
            "count": 135
        }]
    }],
    "page": {
        "size": 4,
        "number": 0,
        "totalElements": 135
    }
}

```

## Creating an Item

A **POST** request is used to create a register item. The response from this request should have a status code of **201 Created** and contain a **Location** header whose value is the URI of the newly created register item. To access the item's details, either perform a **GET** Request on the URI in the **Location** header of the response, or just access the details of the item in the body of the response. We decided to have the API return the created representation in the response to prevent an API consumer from having to hit the API again for the newly created representation.

### Request structure

The JSON body of the POST request adheres to the [Item JSON Schema](#).

### Example request

The following POST request creates a new Item using a JSON representation of `dcat:Dataset`:

```
POST /items HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: localhost
Content-Length: 473

{
  "id" : "urn:example-item-25",
  "uri" : "urn:example-item-25",
  "type" : "dcat:Dataset",
  "title" : "Example Item",
  "description" : "This is an example item",
  "language" : [ "iso639-2:eng" ],
  "characterSet" :
"http://def.seegrid.csiro.au/isotc211/iso19115/2003/code/CharacterSet/utf8",
  "distribution" : [ {
    "title" : "Example Distribution",
    "accessURL" : "http://example.com/example-distribution",
    "type" : "dcat:Distribution"
  } ]
}
```

### Example response

The response returns the URI of the newly created Register Item in the `Location` header with a '201 Created' status code, with the content body containing the newly created item.

```
HTTP/1.1 201 Created
Location: http://localhost:8080/items/6b28163ce062d31533b6f0af1ab3b9e3
Content-Type: application/json
Content-Length: 529

{
  "id" : "6b28163ce062d31533b6f0af1ab3b9e3",
  "uri" : "urn:example-item-25",
  "type" : "dcat:Dataset",
  "title" : "Example Item",
  "description" : "This is an example item",
  "created" : "2016-10-17T19:28:18.848Z",
  "language" : [ "iso639-2:eng" ],
  "characterSet" :
"http://def.seegrid.csiro.au/isotc211/iso19115/2003/code/CharacterSet/utf8",
  "distribution" : [ {
    "title" : "Example Distribution",
    "accessURL" : "http://example.com/example-distribution",
    "type" : "dcat:Distribution"
  } ]
}
```

## Item

An **Item** is the top class of the SRIM hierarchy. It is a sub class of the class `rdf:Resource`, and it represents any information entities or services that can be registered in a register. The **Item resource** is used to retrieve, update, and delete an individual item in the registry service.

### Retrieve a Register Item

A HTTP **GET** Request will retrieve a particular instance of a register item.

#### Query Parameters

None

#### Example request

The following performs a GET Request to retrieve an instance of an item:

```
GET /items/dc6e55caee358f48236facb7608d3945 HTTP/1.1
Accept: application/hal+json
Host: localhost
```

#### Response structure

The response is structured according to the [Item JSON Schema](#).

## Links

The following table shows the link relation types accessible from a register item instance that provide transitions to other states related to this register item instance.

Relation	Description
<a href="#">self</a>	Refers to this <a href="#">item</a> itself
<a href="#">registry:items</a>	Refers to item search endpoint
<a href="#">ogc:jsonld-context</a>	Refers to the JSON context of the item
<a href="#">registry:service</a>	Refers to the root of the registry
<a href="#">curies</a>	Refers to the curies defined for the links

## Example response

===== HAL+JSON Representation

The following response returns the JSON representation of the requested register item instance conforming to the [Register Item JSON Schema](#).

HTTP/1.1 200 OK

Content-Type: application/hal+json

Content-Length: 895

```
{
  "id" : "dc6e55caee358f48236facb7608d3945",
  "uri" : "urn:example-item-22",
  "type" : "dcat:Dataset",
  "title" : "Example Item",
  "description" : "This is an example item",
  "created" : "2016-10-17T19:28:16.583Z",
  "language" : [ "iso639-2:eng" ],
  "characterSet" :
"http://def.seegrid.csiro.au/isotc211/iso19115/2003/code/CharacterSet/utf8",
  "distribution" : [ {
    "title" : "Example Distribution",
    "accessURL" : "http://example.com/example-distribution",
    "type" : "dcat:Distribution"
  } ],
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/items/dc6e55caee358f48236facb7608d3945"
    },
    "registry:items" : {
      "href" : "http://localhost:8080/items"
    },
    "ogc:jsonld-context" : {
      "href" : "http://localhost:8080/context"
    },
    "registry:service" : {
      "href" : "http://localhost:8080"
    }
  }
}
```

===== JSON Representation

The following response returns the JSON representation of the requested register item instance conforming to the [Register Item JSON Schema](#).

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: 529

```
{
  "id" : "b23ef7a5379831ccae0c061a5468a6bb",
  "uri" : "urn:example-item-11",
  "type" : "dcat:Dataset",
  "title" : "Example Item",
  "description" : "This is an example item",
  "created" : "2016-10-17T19:28:12.404Z",
  "language" : [ "iso639-2:eng" ],
  "characterSet" :
"http://def.seegrid.csiro.au/isotc211/iso19115/2003/code/CharacterSet/utf8",
  "distribution" : [ {
    "title" : "Example Distribution",
    "accessURL" : "http://example.com/example-distribution",
    "type" : "dcat:Distribution"
  } ]
}
```

===== Turtle/N3 Representation

The following response returns the Turtle representation of the requested register item instance conforming to the ontology and application profile defining the ItemClass (dcat:Dataset, schema:SchemaMapping).

HTTP/1.1 200 OK

Content-Type: text/n3

Content-Length: 1617

```
@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .
@prefix extent: <http://www.opengis.net/ont/extent#> .
@prefix adms: <http://www.w3.org/TR/vocab-adms/#> .
@prefix pav: <http://purl.org/pav/> .
@prefix org: <http://www.socialml.org/ontologies/organization#> .
@prefix link: <http://www.opengis.net/ont/link#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema-org: <http://schema.org/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix gr: <http://www.heppnetz.de/ontologies/goodrelations/v1#> .
@prefix vcard: <http://www.w3.org/2006/vcard/ns#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix srim: <http://www.opengis.net/ont/testbed12/srim#> .
@prefix locn: <http://www.w3.org/ns/locn#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<urn:example-item-7> a      dcat:Dataset ;
    dct:created      "2016-10-17T15:28:09.767-04:00"^^xsd:dateTime ;
    dct:description  "This is an example item" ;
    dct:language     <http://id.loc.gov/vocabulary/iso639-2/eng> ;
    dct:title        "Example Item" ;
    srim:characterSet
<http://def.seegrid.csiro.au/isotc211/iso19115/2003/code/CharacterSet/utf8> ;
    dcat:distribution [ a      dcat:Distribution ;
                        dct:title  "Example Distribution" ;
                        dcat:accessURL <http://example.com/example-distribution>
                      ] .
```

## Update an Item

A **PUT** request is used to update a Register Item.

A HTTP **PUT** request, with a body containing a JSON representation of the Register Item instance that adheres to [Item JSON Schema](#), is used to update an Item instance. The response of the request returns the HTTP Status Code 204 (no-content) if the update is accepted.

### Request Structure

The JSON request body is structured according to the [Item JSON Schema](#).

### Example Request

The following HTTP request performs a register item update using a JSON representation of the

register item:

```
PUT /items/a22e8128788dc41d7deec94a4d52fe7e HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: localhost
Content-Length: 569

{
  "id" : "a22e8128788dc41d7deec94a4d52fe7e",
  "uri" : "urn:example-item-4",
  "type" : "dcat:Dataset",
  "title" : "New updated title",
  "description" : "This is an example item",
  "created" : "2016-10-17T19:28:05.628Z",
  "language" : [ "iso639-2:eng" ],
  "keyword" : [ "Added keyword" ],
  "characterSet" :
"http://def.seegrid.csiro.au/isotc211/iso19115/2003/code/CharacterSet/utf8",
  "distribution" : [ {
    "title" : "Example Distribution",
    "accessURL" : "http://example.com/example-distribution",
    "type" : "dcat:Distribution"
  } ]
}
```

### Example Response

The response of the request returns the HTTP Status Code 202 (accepted) if the update is accepted, as well as the updated item object so the client can avoid making a second call to get the Item.



```
HTTP/1.1 202 Accepted
Content-Type: application/json
Content-Length: 613
```

```
{
  "id" : "a22e8128788dc41d7deec94a4d52fe7e",
  "uri" : "urn:example-item-4",
  "type" : "dcat:Dataset",
  "title" : "New updated title",
  "description" : "This is an example item",
  "created" : "2016-10-17T19:28:05.628Z",
  "modified" : "2016-10-17T19:28:06.106Z",
  "language" : [ "iso639-2:eng" ],
  "keyword" : [ "Added keyword" ],
  "characterSet" :
"http://def.seegrid.csiro.au/isotc211/iso19115/2003/code/CharacterSet/utf8",
  "distribution" : [ {
    "title" : "Example Distribution",
    "accessURL" : "http://example.com/example-distribution",
    "type" : "dcat:Distribution"
  } ]
}
```

## Delete an Item

A **DELETE** request is used to delete a register item. The deletion will remove the item from the associated register. If the delete operation is successful, the response returns the HTTP code 200 and if the delete operation is unsuccessful, the response returns the HTTP code 404 (Not Found). This functionality should be available only for authorized users.

### Example Request

The following performs a HTTP Delete request on a register item instance URL.

```
DELETE /items/e96a00ecc259d7d9215eb0143462b0e2 HTTP/1.1
Host: localhost
```

### Example Response

If the register item is successfully deleted the response returns the code 200 shown below.

```
HTTP/1.1 200 OK
```

## SPARQL Service

The Semantic Registry Service provides a SPARQL service endpoint, which implements the [SPARQL Protocol](#), that can accept a SPARQL query on the whole registry or a given register. Both HTTP GET

and HTTP POST are supported.

## Query Parameters

	HTTP Method	Query String Parameters	Request Content Type	Request Message Body
query via GET	GET	query (exactly 1)	None	None
query via URL-encoded POST	POST	None	application/x-www-form-urlencoded	* URL-encoded, ampersand-separated query parameters. * query (exactly 1)
query via POST directly	POST	None	application/sparql-query	Unencoded SPARQL query string

## Example Request

The following is an example of a SPARQL query to fetch instances of schema:SchemaMapping with information about source schema and target schema.

```
PREFIX dcat:<http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX schema:<http://www.opengis.net/ont/testbed12/srim/profile/schema#>

SELECT DISTINCT ?schema ?title ?description ?srcSchema ?targetSchema
WHERE {
  ?schema a schema:SchemaMapping;
    dct:title ?title;
    dct:description ?description;
    schema:sourceSchema ?srcSchema;
    schema:targetSchema ?targetSchema
}

LIMIT 4
```

## Response Structure

The SPARQL response conforms to the SPARQL specification. The SPARQL Protocol uses the response status codes defined in HTTP to indicate the success or failure of an operation.

The response body of a successful query operation with a 2XX response is either:

- A SPARQL Results Document in XML, JSON, or CSV/TSV format (for SPARQL Query forms SELECT and ASK); or,
- A RDF graph serialized, in the RDF/XML syntax or an equivalent RDF graph serialization (for SPARQL Query forms DESCRIBE and CONSTRUCT). The content type of the response to a successful query operation must be the media type defined for the format of the response body.

## Example response

The following example shows a SPARQL query response

```
HTTP/1.1 200 OK
Content-Type: application/sparql-results+json
Content-Length: 2352
{
  "head": {
    "vars": [ "schema" , "title" , "description" , "srcSchema" , "targetSchema" ]
  } ,
  "results": {
    "bindings": [
      {
        "schema": { "type": "uri" , "value":
"http://www.opengis.net/testbed12/schemas#MNIS2DDMS" } ,
        "title": { "type": "literal" , "value": "NMIS 2.2 to DDMS SchemaMapping" } ,
        "description": { "type": "literal" , "value": "Schema Mapping from NMIS 2.2
to DDMS" } ,
        "srcSchema": { "type": "uri" , "value":
"http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2" } ,
        "targetSchema": { "type": "uri" , "value":
"http://metadata.dod.mil/mdr/ns/DDMS/2.0" }
      } ,
      {
        "schema": { "type": "uri" , "value":
"http://www.opengis.net/testbed12/schemas#FGDC_TO_ISO_19139Mapping" } ,
        "title": { "type": "literal" , "value": "FGDC to ISO 19139 SchemaMapping" } ,
        "description": { "type": "literal" , "value": "Schema Mapping from
FGDC_STD_001_1998 to ISO 19139" } ,
        "srcSchema": { "type": "uri" , "value":
"https://www.fgdc.gov/schemas/metadata/fgdc-std-001-1998" } ,
        "targetSchema": { "type": "uri" , "value": "http://www.isotc211.org/2005/gmd"
      }
    ]
  }
}
```

## Harvester Service Resources

This section describes the list of resources made accessible by the Harvester Service. The RESTful API has an entry point (service root) which provides links that represent the current state transitions supported by the service.

### Service Root

The root is the entry point of the RESTful Semantic Registry service. When the client consumes the API for the first time, it comes in contact with the root. If the HATEOAS constraint is to be

considered and implemented throughout by clients, then this is the place to start. The root endpoint provides a **set of links** with well-defined relation types that correspond to the supported capabilities of the service. As the Harvester Service API evolves in the future, there may be additional links, each with its own semantics defined by the type of [link relation](#).

The API is considered RESTful as it is fully discoverable **from the root** with **no prior knowledge** – meaning the client should be able to navigate the API by doing a GET on the root. Moving forward, all state changes are driven by the client using the available and discoverable transitions that the REST API provides in representations (hence Representational State Transfer).

### Accessing the Root Endpoint

To access the root of the service and get the list of links, a **GET** request is sent to the base url of the service.

#### Request Structure

This request sends a 'GET' request to the base url of the service. Here the HTTP request returns the list of links supported by the service:

```
GET /harvesters HTTP/1.1
Host: localhost
```

#### Query Parameters

None

#### Response Structure

The request returns a HAL response (application/hal+json) with the **\_links** object that contains links associated with relation types (used as property names in JSON).

Path	Type	Description
<b>_links</b>	<b>Object</b>	The hypermedia links to other states

#### Links

The following table describes the current link relation types supported by the service.

Relation	Description
<b>self</b>	Refers to the registers list itself
<b>harvester:types</b>	Refers to the types of harvesters supported by the registry
<b>harvester:sources</b>	Refers to the harvester sources known to the registry

Relation	Description
<code>harvester:objects</code>	Refers to the harvest objects stored in the registry
<code>harvester:results</code>	Refers to the harvest results available from the registry
<code>registry:service</code>	Refers to the root of the registry
<code>curies</code>	Refers to the curies defined for the links

### Example Response

The following example response shows the current links supported by the service.

```

HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 555

{
  "_links": {
    "self": {
      "href": "http://localhost:8080/harvesters"
    },
    "harvester:types": {
      "href": "http://localhost:8080/harvesters/types"
    },
    "harvester:sources": {
      "href": "http://localhost:8080/harvesters/sources"
    },
    "harvester:objects": {
      "href": "http://localhost:8080/harvesters/objects"
    },
    "harvester:results": {
      "href": "http://localhost:8080/harvesters/results"
    },
    "registry:service": {
      "href": "http://localhost:8080"
    }
  }
}

```

## Harvester Types

A **HarvesterType** defines the specification (class) of harvester for a given source type that produces items.

This resource describes the collection of harvester types supported by the harvester service. They provide a description of the source types that can be harvested and the detailed description of the required and optional parameters used to configure harvester source instance of this type.

## JSON Schema

The JSON representation of the Harvester Type instance has the following fields:

Path	Type	Description	Card.
<code>id</code>	String	Internal identifier for the harvester type. This identifier is referred by harvester source to associate the type of the source with property type.	1
<code>uri</code>	String	URI of the Harvester Type Class (defined in an ontology)	1
<code>type</code>	String	Always 'HarvesterType'.	1
<code>title</code>	String	The title of the harvester	1
<code>description</code>	String	The description of the register	0..1
<code>parameters</code>	Array of Object	Array of Parameter definitions used to configure an instance of this type (called Harvester Source)	0..1
<code>parameters[].name</code>	String	Name of the parameter used in harvester source config as a key.	1
<code>parameters[].title</code>	String	Title of the parameter used for UI display	1
<code>parameters[].description</code>	String	Description of parameter	1
<code>parameters[].datatype</code>	String	Datatype of the parameter. Value can be URI, string, text (for text area), numeric	1
<code>parameters[].optional</code>	boolean	Indicates if the parameter is optional	1
<code>parameters[].allowedValues</code>	Array of String	List of allowed values for a given parameter	0..1
<code>parameters[].default</code>	string	Default value of the parameter if any.	0

## Search Harvester Types

A search for harvester types is done by performing a HTTP **GET** request on the Harvester Types Resource. The harvester type search supports free text search, search by id or uris, and facet values. In addition, it can return results with aggregations on specific facets and customized projection on fields.

### Query Parameters

The following table describes the query parameters that are currently supported

Parameter	Description
<code>q</code>	Text to search for in textual field of the resource.
<code>uri</code>	One or more URIs of the resource to retrieve

Parameter	Description
<code>id</code>	One or more IDs of the resource to retrieve
<code>includeFacet</code>	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.
<code>facet.fieldname</code>	Constraint values of a given facet field name
<code>fields</code>	List of fields to be returned using dot path notation in JSON. This method allows to return custom response controlled by the client with the given fields of interest
<code>pageNumber</code>	The number of the current page (zero-indexed)
<code>pageSize</code>	The count of items on the current page.
<code>sort</code>	The sorting parameters.

### Example request

The following is an example of a GET Request performed on the Harvester Types resource.

```
GET /harvesters/types HTTP/1.1
Host: localhost
```

### Response structure

The response of a Harvester Types request has the following structure:

Path	Type	Description
<code>_embedded</code>	Object	Embedded results
<code>_embedded.registry:harvesterTypes</code>	Array	The array of harvester types results
<code>_embedded.registry:harvesterTypes[].id</code>	String	Identifier for the harvester type
<code>_embedded.registry:harvesterTypes[].type</code>	String	Always 'HarvesterType'
<code>_embedded.registry:harvesterTypes[].title</code>	String	Title of the harvester
<code>_embedded.registry:harvesterTypes[].description</code>	String	Description of the harvester type
<code>aggregations</code>	Array	Aggregation information
<code>page</code>	Array	Information about the current page

Path	Type	Description
<code>_links</code>	Object	The hypermedia links to other states

### Example response

The following example response shows the current links supported by the service.

```
{
  "_embedded": {
    "registry:harvesterTypes": [{
      "id": "csw",
      "type": "HarvesterType",
      "title": "OGC CSW Harvester",
      "description": "OGC Web Catalog Service Harvester",
      "_links": {
        "self": {
          "href": "http://54.208.90.94/registry/harvesters/types/csw"
        }
      }
    }, {
      "id": "dcat",
      "type": "HarvesterType",
      "title": "DCAT Document Harvester",
      "description": "DCAT Harvester type",
      "_links": {
        "self": {
          "href": "http://54.208.90.94/registry/harvesters/types/dcat"
        }
      }
    }, {
      "id": "wfs",
      "type": "HarvesterType",
      "title": "OGC WFS Harvester",
      "description": "Web Feature Service Harvester type",
      "_links": {
        "self": {
          "href": "http://54.208.90.94/registry/harvesters/types/wfs"
        }
      }
    }, {
      "id": "wms",
      "type": "HarvesterType",
      "title": "OGC WMS Harvester",
      "description": "OGC Web Map Service Harvester",
      "_links": {
        "self": {
          "href": "http://54.208.90.94/registry/harvesters/types/wms"
        }
      }
    }
  ]
}
```



```

    }, {
      "id": "pod",
      "type": "HarvesterType",
      "title": "Project Open Data Harvester",
      "description": "Project Open Data file Harvester",
      "_links": {
        "self": {
          "href": "http://54.208.90.94/registry/harvesters/types/pod"
        }
      }
    }, {
      "id": "ckan",
      "type": "HarvesterType",
      "title": "CKAN Harvester",
      "description": "CKAN Harvester",
      "_links": {
        "self": {
          "href": "http://54.208.90.94/registry/harvesters/types/ckan"
        }
      }
    }, {
      "id": "iso19139",
      "type": "HarvesterType",
      "title": "ISO19139 Document Harvester",
      "description": "ISO 19139 Document Harvester",
      "_links": {
        "self": {
          "href": "http://54.208.90.94/registry/harvesters/types/iso19139"
        }
      }
    }
  ]],
  "_links": {
    "self": {
      "href": "http://54.208.90.94/registry/harvesters/types"
    },
    "registry:harvester": {
      "href": "http://54.208.90.94/registry/harvesters"
    },
    "registry:service": {
      "href": "http://54.208.90.94/registry"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/registry/{rel}",
      "name": "registry",
      "templated": true
    }]
  },
  "aggregations": [],
  "page": {
    "size": 20,

```

```
    "totalElements": 7,  
    "totalPages": 1,  
    "number": 0  
  }  
}
```

## Harvester Type

A **HarvesterType** Resource provides the specification of a harvester type. This resource supports only GET operation.

### Retrieve a Harvester Type

Perform a HTTP **GET** Request to retrieve a particular instance of a Harvester Type.

#### Query Parameters

None

#### Example request

The following is an example of a GET Request to retrieve a Harvester Type resource.

```
GET /harvesters/types/csw HTTP/1.1  
Accept: application/hal+json  
Host: localhost
```

#### Response structure

The response is structured according to the [Harvester Type JSON Schema](#).

#### Links

The following table shows the link relation types accessible from a harvester type resource that provide transitions to other states related to the harvester type instance.

Relation	Description
<a href="#">self</a>	Refers to this harvester type itself
<a href="#">harvester:service</a>	Refers to the root of the harvesters endpoint
<a href="#">harvester:types</a>	Refers to the harvester types collection supported by the service
<a href="#">registry:service</a>	Refers to the root of the registry service
<a href="#">curies</a>	Refers to the curies defined for the links

#### Example response

The following example response shows the details of an OGC Web Catalog Harvester Type in

```

HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 827
{
  "id": "csw",
  "uri": "http://www.opengis.net/ont/service/harvesters/types/csw",
  "type": "HarvesterType",
  "title": "OGC CSW Harvester",
  "description": "OGC Web Catalog Service Harvester",
  "parameters": [{
    "name": "requestXML",
    "title": "GetRecords XML Request",
    "description": "GetRecords Request body encoded in XML",
    "datatype": "text",
    "optional": true,
    "default": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<csw:GetRecords\n
service=\"CSW\"\n
version=\"2.0.2\"\n\tstartPosition=\"1\"\n\tmaxRecords=\"50000\"\n
resultType=\"results\"\n  outputFormat=\"application/xml\"\n
outputSchema=\"http://www.isotc211.org/2005/gmd\"\n
xmlns:csw=\"http://www.opengis.net/cat/csw/2.0.2\"\n
xmlns:ogc=\"http://www.opengis.net/ogc\"\n
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"\n
xsi:schemaLocation=\"http://www.opengis.net/cat/csw/2.0.2/CSW-discovery.xsd\"\n
<csw:Query typeNames=\"csw:Record\"\n
<csw:ElementSetName>full</csw:ElementSetName>\n  </csw:Query>\n</csw:GetRecords>"
  }, {
    "name": "resourceType",
    "title": "Resource Type",
    "description": "The type of resource harvested",
    "datatype": "string",
    "optional": true,
    "allowedValues": [
      "http://www.isotc211.org/2005/gmd",
      "urn:ogc:def:ebRIM-AssociationType:OGC:SchemaMapping",
      "urn:ogc:def:ebRIM-ObjectType:OGC:Schema",
      "http://www.w3.org/ns/dcat#Dataset",
      "urn:ogc:def:ebRIM-ObjectType:OGC-I15::DataMetadata"
    ],
    "default": "http://www.isotc211.org/2005/gmd"
  }],
  "_links": {
    "self": {
      "href": "http://localhost:8080/registry/harvesters/csw"
    },
    "harvester:types": {
      "href": "http://localhost:8080/registry/harvesters/types"
    }
  },
}

```

```

    "harvester:service": {
      "href": "http://localhost:8080/registry/harvesters"
    },
    "registry:service": {
      "href": "http://localhost:8080/registry"
    }
  }
}

```

## Harvester Sources

A **HarvesterSource** defines an instance of a harvester type that needs to be harvested. It provides a description of the source that can be harvested. Configuration information includes the values bound to the parameters defined by the harvester type associated with this source, harvesting frequency and register id where harvested item results are saved. The **Harvester Sources Resource** describes the collection of harvester sources currently managed by the harvester service. It is used to create and search harvester sources.

### JSON Schema

The JSON representation of the Harvester Source instance has the following fields:

Path	Type	Description	Card.
id	String	Internal identifier for the harvester, used by the registry	1
type	String	Type of harvester . Refer to the id of the harvester type (for example csw).	1
title	String	The title of the harvester	1
description	String	The description of the register	0..1
created	String	The date of creation (XSD datetime format)	1
modified	String	The date of the last modification (XSD datetime format)	0..1
source	URL	URL of the Source harvest	1
config	Object	Configuration Object defining value binding to the parameter name defined in the Harvester Type (for example 'requestXML' for CSW harvest)	0..1
harvestInterval	String	Harvesting time interval. Use xsd:duration string or MANUAL value (for adhoc harvesting). Default is MANUAL	0..1
registerId	String	Id of register instance that the harvester use to populate	0..1

## Search Harvester Sources

A search for harvester sources is done by performing a HTTP **GET** request on the Harvester Sources Resource. The harvester source search supports free text search, search by id or uris, and facet values. In addition, it can return results with aggregations on specific facets.

### Query Parameters

The following are the query parameters that are currently supported

Parameter	Description
<code>q</code>	Text to search for in textual field of the harvester source.
<code>uri</code>	One or more URIs of the harvester source to retrieve
<code>id</code>	One or more IDs of the harvester source to retrieve
<code>includeFacet</code>	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.
<code>facet.fieldname</code>	Constraint values of a given facet field name
<code>fields</code>	List of fields to be returned using dot path notation in JSON. This method allows to return custom response controlled by the client with the given fields of interest
<code>pageNumber</code>	The number of the current page (zero-indexed)
<code>pageSize</code>	The count of items on the current page.
<code>sort</code>	The sorting parameters.

### Example Request

The following is an example of a GET Request performed on the Harvester Sources resource.

```
GET /harvesters/sources HTTP/1.1
Host: localhost
```

### Response structure

The response of a Harvester Sources request has the following structure in HAL+JSON:

Path	Type	Description
<code>_embedded</code>	<code>Object</code>	Embedded results

Path	Type	Description
<code>_embedded.harvester:sources[]</code>	Array	The array of harvester sources results
<code>_embedded.harvester:sources[].id</code>	String	Identifier for the harvester source
<code>_embedded.harvester:sources[].title</code>	String	Title of the harvester source
<code>_embedded.harvester:sources[].description</code>	String	Description of the harvester source
<code>_embedded.harvester:sources[]._links.self</code>	URL	Link to the instance of this source
<code>aggregations</code>	Array	Aggregation information
<code>page</code>	Array	Information about the current page
<code>_links</code>	Object	The hypermedia links to other states

## Links

The following table shows the link relation types accessible from a harvester sources resource that provide transitions to other states related to the harvester source instance.

Relation	Description
<code>self</code>	Refers to the registers list itself
<code>registry:harvester</code>	Refers to the harvesters landing page
<code>registry:service</code>	Refers to the root of the registry
<code>curies</code>	Refers to the curies defined for the links
<code>first</code>	The first page of results
<code>last</code>	The last page of results
<code>next</code>	The next page of results
<code>prev</code>	The previous page of results

## Example Response

The following example response shows the current links supported by the service.

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 1548

{
```

```

"_embedded": {
  "harvester:sources": [{
    "id": "galdosCSW2",
    "title": "SchemaMapping and Script Harvester from Galdos ebRIM CSW ",
    "description": "This source harvests schemaMapping stored in ebRIM Model",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/registry/harvesters/sources/galdosCSW2"
      }
    }
  }, {
    "id": "envitiaCSW",
    "title": "Testbed12 Envitia ebRIM CSW",
    "description": "Envitia CSW used harvest ebRIM datasets records",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/registry/harvesters/sources/envitiaCSW"
      }
    }
  }, {
    "id": "galdosCSW1",
    "title": "Schema Harvester from Galdos ebRIM CSW ",
    "description": "This source harvests schemas stored in ebRIM Model",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/registry/harvesters/sources/galdosCSW1"
      }
    }
  }, {
    "id": "compusultCSW",
    "title": "Testbed12 Compusult CSW",
    "description": "Compusult CSW used for OGC Testbed 12 to harvest ISO19139
documents",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/registry/harvesters/sources/compusultCSW"
      }
    }
  }, {
    "id": "esriCSW",
    "title": "Testbed12 ESRI CSW",
    "description": "ESRI CSW used for OGC Testbed 12",
    "_links": {
      "self": {
        "href": "http://54.208.90.94/registry/harvesters/sources/esriCSW"
      }
    }
  }
}

```

```

    }],
    "_links": {
      "self": {
        "href": "http://54.208.90.94/registry/harvesters/sources"
      },
      "registry:harvester": {
        "href": "http://54.208.90.94/registry/harvesters"
      },
      "registry:service": {
        "href": "http://54.208.90.94/registry"
      },
      "curies": [{
        "href": "http://www.opengis.net/rels/registry/{rel}",
        "name": "registry",
        "templated": true
      }]
    },
    "aggregations": [],
    "page": {
      "size": 20,
      "totalElements": 5,
      "totalPages": 1,
      "number": 0
    }
  }
}

```

## Creating a Harvester Source

A **POST** request is used to create a harvester source. The response from this request should have a status code of **201 Created** and contain a **Location** header whose value is the URI of the newly created harvester source. To access the harvester source's details, either perform a **GET** Request on the URI in the **Location** header of the response, or just access the details of the harvester source in the body of the response. We decided to have the API return the created representation in the response to prevent an API consumer from having to hit the API again for the newly created representation.

### Request Structure

The following table describes the fields accepted in the JSON request:

Path	Type	Description	Cardinality
<b>id</b>	<b>String</b>	Internal identifier for the harvester, used by the registry. The id can be user defined. If null, the id will be generated by the system.	0..1
<b>type</b>	<b>String</b>	Type of harvester	1



Path	Type	Description	Cardinality
title	String	The title of the harvester	1
description	String	The description of the register	1
source	URL	Source url to harvest	1
config	Object	Configuration Object defining value binding to the parameter name defined in the Harvester Type (for example 'requestXML' for CSW harvest)	0..1
harvestInterval	String	Harvesting time interval. Use xsd:duration string or MANUAL value (for adhoc harvesting). Default is MANUAL	0..1
registerId	String	Id of register instance that the harvester use to populate	0..1

### Example Request

The following POST request creates a new Harvester Source for a CSW Harvester Type.

```
POST /harvesters/sources HTTP/1.1
Content-Type: application/json
Host: localhost
Content-Length: 146

{
  "id" : "example-33",
  "type" : "CSWHarvester",
  "title" : "Example Harvester",
  "description" : "Harvest all Items from Example CSW"
}
```

### Example Response

The response returns the URI of the newly created Harvester Source in the **Location** header with a '201 Created' status code, with the content body containing the newly created harvester source.

HTTP/1.1 201 Created

Location: http://localhost:8080/harvesters/sources/example-33

Content-Type: application/hal+json

Content-Length: 827

```
{
  "id" : "example-33",
  "type" : "CSWHarvester",
  "title" : "Example Harvester",
  "description" : "Harvest all Items from Example CSW",
  "created" : "2016-10-17T19:28:31.601Z",
  "modified" : "2016-10-17T19:28:31.601Z",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/harvesters/sources/example-33"
    },
    "harvester:harvest" : {
      "href" : "http://localhost:8080/harvesters/sources/example-33/harvest"
    },
    "harvester:sources" : {
      "href" : "http://localhost:8080/harvesters/sources"
    },
    "harvester:type" : {
      "href" : "http://localhost:8080/harvesters/types/CSWHarvester"
    },
    "harvester:service" : {
      "href" : "http://localhost:8080/harvesters"
    },
    "registry:service" : {
      "href" : "http://localhost:8080"
    }
  }
}
```

## Harvester Source

A **HarvesterSource** defines an instance of a harvester type to access a given source of information.

### Retrieve a Harvester Source

A HTTP **GET** Request will retrieve a particular instance of Harvester Source.

### Query Parameters

None

### Example Request

The following is an example of a GET Request performed on the Harvester Source resource which accepts HAL+JSON format.

```
GET /harvesters/sources/compusultCSW HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response Structure

The response is structured according to the [Harvester Source JSON Schema](#).

### Links

The following table defines the link relation types accessible from a harvester source resource that provide transitions to other states related to the harvester instance.

Relation	Description
<code>self</code>	Refers to this harvester configuration itself
<code>harvester:harvest</code>	Refers to the link used to harvest items using the harvester configuration
<code>harvester:sources</code>	Refers to the search endpoint for harvester sources
<code>harvester:service</code>	Refers to the root of the harvesters endpoint
<code>harvester:type</code>	Refers to the harvester type definition
<code>registry:service</code>	Refers to the root of the registry
<code>curies</code>	Refers to the curies defined for the links

### Example Response

The following example response is an instance of a CSW Harvester source managed by the service

```

{
  "id": "compusultCSW",
  "type": "csw",
  "title": "Testbed12 Compusult CSW",
  "description": "Compusult CSW used for OGC Testbed 12 to harvest ISO19139
documents",
  "created": "2016-10-03T22:49:27.311Z",
  "modified": "2016-10-03T22:49:27.311Z",
  "source": "http://ogc-testbed12.compusult.net/wes/serviceManagerCSW/csw",
  "config": {
    "resourceType": "http://www.isotc211.org/2005/gmd"
  },
  "harvestInterval": "MANUAL",
  "registerId": "datasets",
  "_links": {
    "self": {
      "href": "http://localhost:8080/registry/harvesters/sources/compusultCSW"
    },
    "harvester:harvest": {
      "href":
"http://localhost:8080/registry/harvesters/sources/compusultCSW/harvest"
    },
    "harvester:sources": {
      "href": "http://localhost:8080/registry/harvesters/sources"
    },
    "harvester:type": {
      "href": "http://localhost:8080/registry/harvesters/types/csw"
    },
    "harvester:service": {
      "href": "http://localhost:8080/registry/harvesters"
    },
    "registry:service": {
      "href": "http://localhost:8080/registry"
    }
  }
}

```

## Update a Harvester Source

A HTTP **PUT** request, with a body containing a JSON representation of the Harvester Source instance, is used to update a Harvester Source instance. The response of the request returns the HTTP Status Code 202 (Accepted) if the update is accepted with the updated Harvester Source description. Both JSON and HAL+JSON formats are accepted as response mime types.

### Request Structure

The following table describes the fields accepted in the JSON request:

Path	Type	Description	Cardinality
id	String	Internal identifier for the harvester, used by the harvester service.	0..1
type	String	Type of harvester	1
title	String	The title of the harvester	1
description	String	The description of the register	1
source	URL	Source url to harvest	1
config	Object	Configuration Object defining value binding to the parameter name defined in the Harvester Type (for example 'requestXML' for CSW harvest)	0..1
harvestInterval	String	Harvesting time interval. Use xsd:duration string or MANUAL value (for adhoc harvesting). Default is MANUAL	0..1
registerId	String	Id of register instance that the harvester use to populate	0..1

### Response Structure

The response is structured according to the [Harvester Source JSON Schema](#).

### Example Request

The following HTTP request performs an update using a JSON representation of the Harvester Source.

```
PUT /harvesters/sources/galdosCSW1 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: localhost
Content-Length: 952

{
  "id": "galdosCSW1",
  "type": "csw",
  "title": "Schema Harvester from Galdos eBRIM CSW ",
```

```

"description": "This source harvests schemas stored in ebRIM Model",
"source": "http://ows.galdosinc.com/indicio/query",
"config": {
  "requestXML": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
    <csw:GetRecords xmlns:env-
ebrim=\"http://www.envitia.com/schemas/georegistry/ebrim-ext\"
xmlns:xmime=\"http://www.w3.org/2005/05/xmlmime\"
xmlns:dct=\"http://purl.org/dc/terms/\"
xmlns:csw=\"http://www.opengis.net/cat/csw/2.0.2\"
xmlns:gml=\"http://www.opengis.net/gml\"
xmlns:wrs=\"http://www.opengis.net/cat/wrs/1.0\"
xmlns:ows=\"http://www.opengis.net/ows\"
xmlns:ogc=\"http://www.opengis.net/ogc\"
xmlns:dc=\"http://purl.org/dc/elements/1.1/\"
xmlns:xlink=\"http://www.w3.org/1999/xlink\"
service=\"CSW\"
version=\"2.0.2\" \r\n\tresultType=\"results\" outputSchema=\"urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0\"
startPosition=\"1\" maxRecords=\"50\">
  <csw:Query
typeNames=\"wrs:ExtrinsicObject\">
  <csw:ElementSetName>full</csw:ElementSetName>
    <csw:Constraint version=\"1.1.0\">
      <ogc:Filter>
        <ogc:PropertyIsEqualTo>
<ogc:PropertyName>@objectType</ogc:PropertyName>
<ogc:Literal>urn:ogc:def:ebRIM-ObjectType:OGC:Schema</ogc:Literal>
          </ogc:PropertyIsEqualTo>
        </ogc:Filter>
      </csw:Constraint>
    </csw:Query>
  </csw:GetRecords>\",
  "resourceType": "urn:ogc:def:ebRIM-ObjectType:OGC:Schema"
},
"harvestInterval": "MANUAL",
"registerId": "schemas"
}

```

### Example Response

The response returns the HTTP Status Code 202 (accepted) if the update is accepted, and the updated harvester source is returned so the client can avoid making a second call to get the harvester source. The response of the request conforms to the [Harvester Source JSON Schema](#).

HTTP/1.1 202 Accepted

Content-Type: application/json;charset=UTF-8

Content-Length: 980

```
{
  "id": "galdosCSW1",
  "type": "csw",
  "title": "Schema Harvester from Galdos ebRIM CSW ",
  "description": "This source harvests schemas stored in ebRIM Model",
  "created": "2016-10-03T22:49:27.542Z",
  "modified": "2016-10-03T22:49:27.542Z",
  "source": "http://ows.galdosinc.com/indicio/query",
  "config": {
    "requestXML": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
      <csw:GetRecords xmlns:env-
ebrim=\"http://www.envitia.com/schemas/georegistry/ebrim-ext\"
xmlns:xmime=\"http://www.w3.org/2005/05/xmlmime\"
xmlns:dct=\"http://purl.org/dc/terms/\"
xmlns:csw=\"http://www.opengis.net/cat/csw/2.0.2\"
xmlns:gml=\"http://www.opengis.net/gml\"
xmlns:wrs=\"http://www.opengis.net/cat/wrs/1.0\"
xmlns:ows=\"http://www.opengis.net/ows\"
xmlns:ogc=\"http://www.opengis.net/ogc\"
xmlns:dc=\"http://purl.org/dc/elements/1.1/\"
xmlns:xlink=\"http://www.w3.org/1999/xlink\"
service=\"CSW\"
version=\"2.0.2\" \r\n\tresultType=\"results\" outputSchema=\"urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0\"
startPosition=\"1\" maxRecords=\"50\">
      <csw:Query
typeNames=\"wrs:ExtrinsicObject\">
      <csw:ElementSetName>full</csw:ElementSetName>
      <csw:Constraint version=\"1.1.0\">
        <ogc:Filter>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>@objectType</ogc:PropertyName>
            <ogc:Literal>urn:ogc:def:ebRIM-ObjectType:OGC:Schema</ogc:Literal>
          </ogc:PropertyIsEqualTo>
        </ogc:Filter>
      </csw:Constraint>
    </csw:Query>
    </csw:GetRecords>",
    "resourceType": "urn:ogc:def:ebRIM-ObjectType:OGC:Schema"
  },
}
```

```
"harvestInterval": "MANUAL",  
"registerId": "schemas"  
}
```

## Delete a Harvester Source

A **DELETE** request is used to delete a harvester source. If the delete operation is successful, the response returns the HTTP code 200 and if the delete operation is unsuccessful, the response returns the HTTP code 404 (Not Found). This functionality should be available only for authorized users.

### Example Request

The following example performs a HTTP Delete request on a harvester source URL.

```
DELETE /harvesters/sources/example-33 HTTP/1.1  
Host: localhost
```

### Example Response

If the harvester source is successfully deleted the response return the code 200 shown below.

```
HTTP/1.1 200 OK
```

## Harvest Action Resource

A Harvest Action is an "Action" Resource that triggers the harvesting of a source defined by a Harvester Source instance. This operation is invoked by a HTTP Get Operation. The current implementation performs only synchronous operation, but future extension will support asynchronous operations.

### Harvest a Source

#### Query Parameters

None

#### Example Request

The following is an example of a GET Request to perform a harvest action that accepts HAL+JSON format.

```
GET /harvesters/sources/galdosCSW1/harvest HTTP/1.1  
Accept: application/hal+json  
Host: localhost
```



## Response Structure

The response of a Harvest action request in the synchronous model has the following structure in HAL+JSON:

Path	Type	Description	Card,
harvesterSource	Object	Harvester Source object used as defined in the <a href="#">Harvester Source JSON Schema</a>	1
items[]	Array	The array of items collected	1
items[].id	String	The identifier of the harvested item	1
items[].uri	String	The uri identifier of the harvested item	0..1
items[].type	String	The type (item class id) of the harvested item	1
items[].title	String	The title of the harvested item	1
startTime	XSD dateTime	Start time of harvesting	1
endTime	xsd dateTime	End time of harvesting	1
duration	integer	Duration of harvesting in milliseconds	1
recordMatched	integer	Number of records matched	1
recordHarvested	integer	Number of records harvested and successfully stored	1
recordsFailed	integer	Number of records failed	1
_links	Object	The hypermedia links to other states (if using HAL)	0.. 1

## Example response

The following is the response to the harvest action in HAL+JSON

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 2703
{
```

```

"harvesterSource": {
  "id": "galdosCSW1",
  "type": "csw",
  "title": "Schema Harvester from Galdos ebRIM CSW ",
  "description": "This source harvests schemas stored in ebRIM Model",
  "created": "2016-11-08T02:51:43.747Z",
  "modified": "2016-11-08T02:51:43.747Z",
  "source": "http://ows.galdosinc.com/indicio/query",
  "config": {
    "requestXML": "<?xml version=\\"1.0\\" encoding=\\"UTF-
8\\"?>\r\n<csw:GetRecords xmlns:env-
ebRim=\\"http://www.envitia.com/schemas/georegistry/ebRim-
ext\\" \r\n\txmlns:xmime=\\"http://www.w3.org/2005/05/xmlmime\\"
xmlns:dct=\\"http://purl.org/dc/terms/\\" \r\n\txmlns:csw=\\"http://www.opengis.net/cat/cs
w/2.0.2\\"
xmlns:gml=\\"http://www.opengis.net/gml\\" \r\n\txmlns:wrs=\\"http://www.opengis.net/cat/w
rs/1.0\\"
xmlns:ows=\\"http://www.opengis.net/ows\\" \r\n\txmlns:ogc=\\"http://www.opengis.net/ogc\\"
xmlns:dc=\\"http://purl.org/dc/elements/1.1/\\" \r\n\txmlns:xlink=\\"http://www.w3.org/19
9/xlink\\" service=\\"CSW\\" version=\\"2.0.2\\" \r\n\tresultType=\\"results\\"
outputSchema=\\"urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0\\" \r\n\tstartPosition=\\"1\\"
maxRecords=\\"50\\">\r\n\t<csw:Query
typeNames=\\"wrs:ExtrinsicObject\\">\r\n\t\t<csw:ElementSetName>full</csw:ElementSetName
>\r\n\t\t<csw:Constraint
version=\\"1.1.0\\">\r\n\t\t\t<ogc:Filter>\r\n\t\t\t\t<ogc:PropertyIsEqualTo>\r\n\t\t\t\t\t
\t\t<ogc:PropertyName>@objectType</ogc:PropertyName>\r\n\t\t\t\t\t\t<ogc:Literal>urn:ogc:
def:ebRIM-
ObjectType:OGC:Schema</ogc:Literal>\r\n\t\t\t\t\t\t</ogc:PropertyIsEqualTo>\r\n\t\t\t\t\t</ogc
:Filter>\r\n\t\t\t</csw:Constraint>\r\n\t\t</csw:Query>\r\n</csw:GetRecords>",
    "resourceType": "urn:ogc:def:ebRIM-ObjectType:OGC:Schema"
  },
  "harvestInterval": "MANUAL",
  "registerId": "schemas"
},
"items": [{
  "id": "f981d111fdbbf8e7916f6dfe9dcc7c69",
  "uri": "urn:uuid:566ff231-e768-41de-bb26-e0def91ceb9b",
  "type": "schema:Schema",
  "title": "National System for Geospatial Intelligence Metadata Implementation
Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2.0"
}, {
  "id": "315a1576cd09615683160f9f1e84ca97",
  "uri": "urn:uuid:02200156-4271-4d52-a2e1-47d268680aa7",
  "type": "schema:Schema",
  "title": "Department of Defense Discovery Metadata Specification (DDMS),
Version 2.0"
}],
"startTime": "2016-11-08T06:48:12.443Z",
"endTime": "2016-11-08T06:48:20.250Z",
"duration": 7807,
"recordsMatched": 2,

```

```
"recordsHarvested": 2,  
"recordsFailed": 0,  
"_links": {  
  "harvester:source": {  
    "href": "http://localhost:8080/harvesters/sources/galdosCSW1"  
  },  
  "harvester:service": {  
    "href": "http://localhost:8080/harvesters"  
  }  
}  
}
```

h

# Appendix E: Semantic Mediation Service REST API

## Overview

This document describes the initial version of RESTful Semantic Mediation Service API. We anticipate that a variety of clients may be using the Semantic Mediation Service, and as a consequence it is difficult to accommodate the needs of every type of client. It is almost certain that the REST API will evolve and be modified as more requirements and features are added to the service. The hypermedia-driven API provides a robust approach to evolve the API without breaking the client ecosystem, as long as the clients are using the semantics of the link relation types. For this reason, the service has adopted a hypermedia-driven RESTful API by default, which meets level 3 of the [Richardson Maturity Model](#).

The default serialization of the information model in the service is JSON, as it is understood by most programming languages. However, to support machine-processable information, we enforce the JSON to be compatible Linked data by using JSON-LD Context. In this way, data can be converted to Linked Data Representation and be reasoned on and linked to other information expressed as Linked Data. The REST API does support content negotiation to return a response in Linked Data Format (RDF/XML, Turtle, NTriple) when applicable. The REST API in this document provides the core minimum functionalities that are based on the Schema SRIM Application Profile.

Creation/Retrieval/Deletion operations are optional for this service. We choose to delegate these operations to the Semantic Registry Service implementing the SRIM Schema Application Profile. For more information about these operations, refers to the documentation of the Semantic Registry API. The Semantic Mediation Service can delegate the operations of search of schemas and schema mappings (as demonstrated by the implementation in the testbed). However the response using hypermedia links is modified to provide links that are relevant for mediation operations. For example a schema mapping retrieval will provide hypermedia links to its source schema and target schema. A schema may provide a link to all the schema mappings using this schema as a source or target.

### NOTE

The example URL used in this documentation used the hostname <http://localhost:8080>. This hostname should be replaced by the entry point (baseURL) of the service that you want to access in the format: <http://{hostname}:{port}/{rootPath}>

## HTTP verbs

The RESTful API tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs. The following HTTP verbs are used by the Service.

Verb	Usage
GET	Used to retrieve a resource

Verb	Usage
POST	Used to create a new resource
PUT	Used to update an existing resource with a complete update
DELETE	Used to delete an existing resource

## HTTP status codes

The Service REST API tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP status codes. The following table summarizes the status codes and usage.

Status code	Usage
200 OK	The request completed successfully
201 Created	A new resource has been created successfully. The resource's URI is available from the response's <code>Location</code> header
204 No Content	An update to an existing resource has been applied successfully
400 Bad Request	The request was malformed. The response body will include an error providing further information
404 Not Found	The requested resource did not exist
405 Method not allowed	The request was made of a resource using a request method not supported by that resource; for example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.
409 Conflict	The request could not be completed due to a conflict with the current state of the resource. This code is only allowed in situations where it is expected that the user might be able to resolve the conflict and resubmit the request.
415 Unsupported media type	The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.
500 Internal Server Error	The Web server encountered an unexpected condition that prevented it from fulfilling the request by the client

## Headers

Every response has the following header(s):

Name	Description
Content-Type	The Content-Type of the payload, e.g. <code>application/hal+json</code>

Future headers may be added for managing access control to the resources managed by the service.

## Errors

Whenever an error response (status code  $\geq 400$ ) is returned, the body will contain a JSON object that describes the problem. The error object has the following structure:

Path	Type	Description
timestamp	Number	The time, in milliseconds, at which the error occurred
status	Number	The HTTP status code, e.g. <code>400</code>
error	String	The HTTP error that occurred, e.g. <code>Bad Request</code>
message	String	A description of the cause of the error
path	String	The path to which the request was made

For example, a request that attempts to apply a non-existent resource to a register item will produce a `400 Bad Request` response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Content-Length: 189

{
  "timestamp" : 1461855421813,
  "status" : 400,
  "error" : "Bad Request",
  "message" : "The resource item 'http://localhost:8080/schemas/123' does not exist",
  "path" : "/schemas"
}
```

## Paging and Sorting

### Paging

Rather than return everything from a large result set, the REST API recognizes some URL parameters that will influence the page size (`size` parameter) and starting page number (`page` parameter) as well as sorting of the result set (`sort` parameter).

See the following example, where we set the page size to 5 and request the third page (page 2) as page numbers are zero-indexed:

```
GET schemas?page=2&size=5 HTTP/1.1
Host: localhost
```

The paginated results in HAL format returns the following response:

```
{
  "_embedded": {
    ...data...
  },
  "_links": {
    "self": {
      "href": "http://localhost:8080/schemas"
    },
    "first": {
      "href": "http://localhost:8080/schemas?page=0&size=5"
    },
    "prev": {
      "href": "http://localhost:8080/schemas?page=1&size=5"
    },
    "next": {
      "href": "http://localhost:8080/schemas?page=3&size=5"
    },
    "last": {
      "href": "http://localhost:8080/schemas?page=5&size=5"
    },
  },
  "page": {
    "size": 5,
    "totalElements": 27,
    "totalPages": 6,
    "number": 2
  }
}
```

Each paged response will return links to the first, previous, next, and last page of results based on the current page using the IANA defined link relations `first`, `prev`, `next`, `last`. If you are currently at the first page of results, however, no `prev` link will be rendered. The same is true for the last page of results: no `next` link will be rendered.

The paginated results also have extra data about the page settings , including the size of a page, total elements, total pages, and the page number you are currently viewing. This extra information makes it very easy for the consumer to configure UI tools like sliders or indicators to reflect the overall position the user is in viewing the data. For example, the document above shows we are

looking at the third page (with page numbers indexed to 0 being the first).

## Sorting

The REST API recognizes sorting parameters. To have your results sorted on a particular property, add a `sort` URL parameter with the name of the property you want to sort the results on. You can control the direction of the sort by appending a `,` to the the property name plus either `asc` or `desc`.

The following examples will sort results by title in ascending order:

```
GET /schemas?sort=title,asc HTTP/1.1
Host: www.mydomain.com
```

To sort the results by more than one property, keep adding as many `sort=PROPERTY` parameters as you need. They will be added in the order they appear in the query string.

## Search Results

A number of endpoints of the service return search results. They usually supports Level 2 (JSON) and Level 3 response (HAL+JSON). The search results contains the collection of matched items, paging information and optionally faceted aggregation results. The following describes the response structure for each format.

### HAL+JSON Search Results

Path	Type	Description
<code>_embedded</code>	Array	The HAL <code>_embedded</code> field that contains a collection of instances
<code>embedded._collectionName[]</code>	Array	The array of items instances defined a given JSON schema. The <code>collectionName</code> may varied depending of the types of items contained in the collection.
<code>aggregations[]</code>	Array	The aggregation results as defined in the <a href="#">Aggregation JSON Schema</a> . This field is optional is no faceted search is performed on the endpoint.
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation



Path	Type	Description
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket
<code>_links</code>	Object	Links to other states
<code>page</code>	Object	The page state. See <<_paging,Paging section>
<code>page.size</code>	Number	The page size
<code>page.totalElements</code>	Number	The total elements matched by the search request
<code>page.totalPages</code>	Number	The total number of pages in the results
<code>page.number</code>	Number	The current page number (starts at 0)
<code>_links[]</code>	Array	Array of hypermedia links to other reachable states.

## JSON Search results

When the JSON or JSON-LD response is retrieved, the matched items are placed in an array referred by the *results* field. Aggregations results are present only faceted search is performed and paging information are returned.

Path	Type	Description
<code>results[]</code>	Array	The array of vocabulary instances that matches the search criteria
<code>aggregations[]</code>	Array	The aggregation results as defined in the <a href="#">Aggregation JSON Schema</a> . This field is optional is no faceted search is performed on the endpoint.
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation

Path	Type	Description
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket
<code>page</code>	Object	The page state. See <<_paging,Paging section>
<code>page.size</code>	Number	The page size
<code>page.totalElements</code>	Number	The total elements matched by the search request
<code>page.totalPages</code>	Number	The total number of pages in the results
<code>page.number</code>	Number	The current page number (starts at 0)

### Aggregation JSON Schema

In many use cases, it is useful to aggregate the search results according some facets values (number of items per topic, per publisher, etc). Each facet is composed of a name, global metrics (right now only total count is supported), a set of buckets containing a label (unique value of the facet) and the total count for each label within the context of the facet and search results. The JSON schema of the Aggregation results is defined in the following table.

Path	Type	Description
<code>aggregations[]</code>	Array	The aggregation results
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket

### Resources Summary

The following resources are the core resources supported by the Semantic Mediation Service.

Resources	Description	Operations
Capabilities	This resource describes the capabilities of the service	GET
Schema Mapping Collection	This resource represents a collection of schema mappings and supports creation of schema mappings and search on a collection of schema mappings based on search criteria	GET, POST (optional)
Schema Mapping	This resource supports retrieval, update, and deletion of schema mapping instance	GET, PUT (optional),DELETE (optional)
Schema Collection	represents a collection of schemas and supports creation of schema and search on a collection of schemas based on search criteria	GET, POST (optional)
Schema	This resource supports retrieval update, and deletion of schema instance	GET, PUT (optional), DELETE (optional)
JSONLD Context	This resource returns the JSON-LD context associated with the JSON representation returned by the service	GET
Validator	This resource performs validation of a data against a given schema	GET,POST
Transformer	This resource performs transformation of data from one schema to another	GET,POST

## Link relation types

The following table describes the list of link relation types used by the Semantic Mediation Service. We use the OGC namespace to define the relationships in order to favor reusability across all OGC service specifications.

Relation type	URI	Description
mediation:capabilities	<a href="http://www.opengis.net/rels/mediation/capabilities">http://www.opengis.net/rels/mediation/capabilities</a>	Reference to the capabilities of the service
mediation:schemas	<a href="http://www.opengis.net/rels/mediation/schemas">http://www.opengis.net/rels/mediation/schemas</a>	Reference to the schema collection
mediation:sourceSchema	<a href="http://www.opengis.net/rels/mediation/sourceSchema">http://www.opengis.net/rels/mediation/sourceSchema</a>	Reference to the source schema of a schema mapping
mediation:targetSchema	<a href="http://www.opengis.net/rels/mediation/targetSchema">http://www.opengis.net/rels/mediation/targetSchema</a>	Reference to the target schema of a schema mapping
mediation:schemaMappings	<a href="http://www.opengis.net/rels/mediation/schemaMappings">http://www.opengis.net/rels/mediation/schemaMappings</a>	Reference to the schema mapping collection

Relation type	URI	Description
mediation:sourceSchemaMappings	<a href="http://www.opengis.net/rels/mediation/sourceSchemaMappings">http://www.opengis.net/rels/mediation/sourceSchemaMappings</a>	Reference to the schema mapping collection
mediation:targetSchemaMappings	<a href="http://www.opengis.net/rels/mediation/targetSchemaMappings">http://www.opengis.net/rels/mediation/targetSchemaMappings</a>	Reference to the schema mapping collection
ogc:jsonldContext	<a href="http://www.opengis.net/rels/jsonldContext">http://www.opengis.net/rels/jsonldContext</a>	Reference to the JSON-LD context to apply to the JSON content to transform it to Linked Data
mediation:validator	<a href="http://www.opengis.net/rels/mediation/validator">http://www.opengis.net/rels/mediation/validator</a>	Reference to the validator service
mediation:transform	<a href="http://www.opengis.net/rels/mediation/transform">http://www.opengis.net/rels/mediation/transform</a>	Reference to the schema mapping transformer service
ogc:documentation	<a href="http://www.opengis.net/rels/restdoc">http://www.opengis.net/rels/restdoc</a>	Reference to the REST documentation of the service

## Content negotiation

Most of the resource can serve multiple representations including:

Format	Mime type	Description
HAL+JSON	application/hal+json	It is the <b>default format</b> of the service. The JSON payload is compatible with JSON-LD and is aligned with the Schema SRIM Application Profile Ontology.
JSON-LD	application/ld+json	Compliant with the Schema SRIM Application Profile Ontology using the JSON-LD context.
RDF/XML	application/rdf+xml	Compliant with the Schema SRIM Application Profile Ontology
Turtle	text/turtle	Compliant with the Schema SRIM Application Profile Ontology
N-Triples	text/ntriples	Compliant with the Schema SRIM Application Profile Ontology

## Resources descriptions

### Service Entry Point

The entry point (also called root) of the RESTful Semantic Mediation service is what the client comes into contact with when consuming the API for the first time. If the HATEOAS constraint is to be considered and implemented throughout by clients, then this is the place to start. The root

endpoint provides a **set of links** with well-defined relation types that corresponds to the supported capabilities of the service. As the Semantic Mediation Service API evolves in the future, there would be many more links, each with it's own semantics defined by the type of [link relation](#).

The API is considered as RESTful as it is fully discoverable **from the root** and with **no prior knowledge** – meaning the client should be able to navigate the API by doing a GET on the root. Moving forward, all state changes are driven by the client using the available and discoverable transitions that the REST API provides in representations (hence Representational State Transfer).

### Accessing the root endpoint

To get access to the root of the service and get the list of links, a **GET** request is sent to the base url of the service.

#### Request structure

This request sends a 'GET' request to the base url of the service. Here the HTTP request associated to get the list of links supported by the service:

```
GET / HTTP/1.1
Host: localhost
```

The only representation is HAL+JSON.

#### Query Parameters

None

#### Response structure

The response is returning a HAL response (application/hal+json) with the **\_links** object that contains links associated with relation types (used as property names in JSON).

Path	Type	Description
@context	String	<a href="#">JSON-LD Context</a> applicable to JSON response of the service
type	String	The type of the resource (always Service)
title	String	Title of the service
description	String	Description of the service
categories	Array	Categories of the service
_links	Object	<a href="#">Links</a> to other resources

#### Links

The following table describes the current link relation types supported by the service.

Relation	Description
ogc:capabilities	Refers to the <a href="#">capabilities</a> of this service.
ogc:schemas	Refers to the <a href="#">schemas</a> supported by this service
ogc:schemaMappings	Refers to the <a href="#">schema mappings</a> managed by this service
ogc:validator	Refers to the validator service
ogc:transform	Refers to transformer service
ogc:jsonContext	Refers to JSON Context of the service
curies	The curies to use to expand the link relation types to URI

### Example response

The following example response shows the current links supported by the service.

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 921
```

```
{
  "@context" : "http://localhost:8080/context",
  "type" : "http://www.opengis.net/ont/srim#Service",
  "title" : "Semantic Mediation Service",
  "description" : "Semantic Mediation Service",
  "categories" : [ "http://www.opengis.net/specs/testbed/sms" ],
  "_links" : {
    "ogc:capabilities" : {
      "href" : "http://localhost:8080/capabilities"
    },
    "ogc:schemas" : {
      "href" : "http://localhost:8080/schemas"
    },
    "ogc:schemaMappings" : {
      "href" : "http://localhost:8080/mappings"
    },
    "ogc:jsonContext" : {
      "href" : "http://localhost:8080/context"
    },
    "ogc:transform" : {
      "href" : "http://localhost:8080/transform"
    },
    "ogc:validator" : {
      "href" : "http://localhost:8080/validate"
    },
    "curies" : [ {
      "href" : "http://www.opengis.net/rels/{rel}",
      "name" : "ogc",
      "templated" : true
    } ]
  }
}
```

## Capabilities

This resource describes the capabilities of the service, including the supported schema and schema mapping languages, application profiles, transformers and formats.

### Query Parameters

There is no query parameter to retrieve the capabilities.

### Example request

The following is an example of GET Request performed on the Capability resource.

```
GET /capabilities HTTP/1.1
Accept: application/hal+json
Host: localhost
```

## Response structure

The response of Capability request has the following structure:

Path	Type	Description
<code>schemaLanguages[]</code>	Array	schema Languages supported by the service
<code>schemaLanguages[].uri</code>	String	The URI of the schema language
<code>schemaLanguages[].type</code>	String	The RDF type of the schema language (always <code>skos:Concept</code> )
<code>schemaLanguages[].prefLabel</code>	String	The human readable prelabel of this schema language
<code>schemaLanguages[].notation</code>	String	The notation (abbreviation) of the schema language
<code>schemaMappingLanguages[]</code>	Array	schema Languages supported by the service
<code>schemaMappingLanguages[].uri</code>	String	The URI of the schema language
<code>schemaMappingLanguages[].type</code>	String	The RDF type of the schema mapping language (always <code>skos:Concept</code> )
<code>schemaMappingLanguages[].prefLabel</code>	String	The human readable prelabel of this schema mapping language
<code>schemaMappingLanguages[].notation</code>	String	The notation (abbreviation) of the schema mapping language
<code>_links</code>	Object	The hypermedia links to other states

## Links

The following table describes the current link relation types in the HAL capabilities response

Relation	Description
<code>service</code>	Link to the root of the service
<code>self</code>	Refers to this endpoint.
<code>curies</code>	Refers to the curies defined for the links



## Example response

The following example response shows a capability of the service is represented.

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 30456

{
  "schemaLanguages": [{
    "type": "skos:Concept",
    "prefLabel": "XML Schema",
    "uri": "http://www.w3.org/TR/xmlschema-1/",
    "notation": "XMLSchema"
  }, {
    "type": "skos:Concept",
    "prefLabel": "Web Ontology Language Full/DL/Lite",
    "uri": "http://www.opengis.net/ont/testbed12/srim/profile/schema#OWL",
    "notation": "OWL"
  }, {
    "type": "skos:Concept",
    "prefLabel": "Resource Description Framework Schema",
    "uri": "http://www.opengis.net/ont/testbed12/srim/profile/schema#RDFSschema",
    "notation": "RDFSschema"
  }, {
    "type": "skos:Concept",
    "prefLabel": "Relax NG",
    "uri": "http://www.opengis.net/ont/testbed12/srim/profile/schema#RELAXNG",
    "notation": "RelaxNG"
  }, {
    "type": "skos:Concept",
    "prefLabel": "Simple Knowledge Organization System",
    "uri": "http://www.opengis.net/ont/testbed12/srim/profile/schema#SKOS",
    "notation": "SKOS"
  }
],
  "schemaMappingLanguages": [{
    "type": "skos:Concept",
    "prefLabel": "eXtensible StyleSheet Transform",
    "uri": "https://www.w3.org/TR/xslt20/",
    "notation": "XSLT"
  }
],
  "_links": {
    "self": {
      "href": "http://localhost:8083/capabilities"
    },
    "service": {
      "href": "http://localhost:8083"
    }
  }
}
```

## JSON-LD Context

The JSON produced by the Semantic Mediation Service is compatible with the Schema SRIM Application Profile, by using JSON-LD context. The context is made accessible through an endpoint, so it can be referred and imported by JSON-LD processor to be converted into a Linked Data representation adhering to the Schema SRIM Profile.

### Query Parameters

There is no query parameter to retrieve the JSON-LD context.

### Example request

The following is an example of GET Request performed on the JSON-LD context resource.

```
GET /context HTTP/1.1
Host: localhost
```

### Response structure

The response of JSON-LD Context is conforming to the standard JSON-LD.

Path	Type	Description
@context	Object	<a href="#">JSON-LD Context</a> applicable to JSON response of the service

### Example response

The following example response shows a JSON-LD context of the service.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2222

{
  "@context": {
    "pav": "http://purl.org/pav/",
    "dct": "http://purl.org/dc/terms/",
    "owl": "http://www.w3.org/2002/07/owl#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "srim": "http://www.opengis.net/ont/testbed/12/srim#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "dcat": "http://www.w3.org/ns/dcat#",
    "iso639-2": "http://id.loc.gov/vocabulary/iso639-2/",
    "lingvoj": "http://www.lingvoj.org/ontology#",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "ldp": "http://www.w3.org/ns/ldp#",
```

```

"type": "@type",
"uri": "@id",
"value": "@value",
"lang": "@language",
"results": "ldp:contains",
"title": "dct:title",
"titleMap": {
  "@id": "dct:title",
  "@container": "@language"
},
"description": "dct:description",
"descriptionMap": {
  "@id": "dct:description",
  "@container": "@language"
},
"category": {
  "@id": "dct:type",
  "@type": "@id"
},
"hasVersion": "dct:hasVersion",
"currentVersion": {
  "@id": "pav:currentVersion",
  "@type": "@id"
},
"version": "pav:version",
"creator": "dct:creator",
"contributor": "dct:contributor",
"publisher": "dct:publisher",
"language": "dct:language",
"iso2Code": "lingvoj:iso1",
"iso3Code": "lingvoj:iso2",
"notation": "skos:notation",
"label": "rdfs:label",
"name": "foaf:name",
"sameAs": {
  "@id": "owl:sameAs",
  "@type": "@id"
},
"distributions": "dcat:distribution",
"license": "dct:license",
"issued": "dct:issued",
"modified": "dct:modified",
"prefix": "wov:prefix",
"namespace": "wov:namespace",
"releaseNote": {
  "@id": "wov:releaseNote",
  "@container": "@language"
},
"accessURL": "dcat:accessURL",
"mediaType": "dcat:mediaType"

```

```
}  
}
```

## Schema Collection

A **Schema** defines the vocabulary and structure of data. The **Schema Collection resource** represents a collection of Schema instances. It is used to create and search schemas.

## JSON Schema

As the registry service can support different item classes and application profiles, the JSON schema supported by the service is not unique. However there is a set of core properties that will always be present for any items in the registry. The following table describes the core minimal properties for a item.

Path	Type	Description	Card.
<code>id</code>	<code>String</code>	Internal identifier for the item, used by the registry	1
<code>uri</code>	<code>String</code>	Linked Data URI for the item (equivalent to <code>@id</code> in JSON-LD)	0..1
<code>type</code>	<code>String</code>	The Item class identifier as published in the capability.	1
<code>title</code>	<code>String</code>	The title of the item	1
<code>titleMap</code>	<code>Object</code>	The title map for each language of the title . Each key corresponds to the two letter language identifier name defined in the capabilities ( for example "en")	0..1
<code>description</code>	<code>String</code>	The description of the item	0..1
<code>descriptionMap</code>	<code>Object</code>	The description map for each language of the description. Each key corresponds to the two letter language identifier name defined in the capabilities ( for example "en")	0..1

Path	Type	Description	Card.
created	String	The date of creation (XSD datetime format) (generated by the service)	1
modified	String	The date of the last modification (XSD datetime format) (generated by the service)	0..1
namespace	URI	This property store the namespace of the schema when available	0..1
schemaLanguage	URI	This property holds the language used in the schema if applicable.	1

### Search schemas

A search for schemas is performed by performing a HTTP **GET** request. The schema search supports free text, by ids, by uris, CQL constraint and can return results with aggregations on specific facets.

### Query Parameters

The current parameters supported by the GET request are for paging and sorting.

The following query parameters are supported in the query:

Parameter	Description
q	Text to search in textual fields
uri	One or more URIs of schema instances
id	One or more id of schema instances
includeFacet	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.
facet.fieldname	Constraint values of a given facet field name
constraint	A constraint expressed in CQL. This is used to express more advanced query filtering
fields	One or more fields to be included in the response. Use JSON path dot notation for referring paths (ex: publisher.name)

Parameter	Description
pageNumber	The number of the current page as defined in <a href="#">Paging section</a>
pageSize	The count of items on the current page as defined in <a href="#">Paging section</a>
sort	The sorting parameters as defined in <a href="#">Sorting section</a> .

### Example request

The following is an example of GET Request performed on the Schemas resource to the first page composed of 3 schemas and sorted by title in ascending order.

```
GET /schemas HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The following fields are present in the response and adhere to the [JSON Schema definition](#).

### Links

The following link relation types are provided in the response to allow the transition to others states from the schemas collection embedded in the response.

Relation	Description
self	Refers to this search request itself
service	Refers to the mediation service
curies	Refers to the curies defined for the links

### Example response

The embedded objects in the response of the request conforms to the [Schema JSON Schema](#). The rest of the response return paging information and links to other states.

```
{
  "_embedded": {
    "mediation:schemas": [{
      "id": "643d2431ff31955de485f84cbbc598d1",
      "uri": "https://www.fgdc.gov/schemas/metadata/fgdc-std-001-1998",
      "type": "schema:Schema",
      "title": "Content Standards for Digital Geospatial Metadata Version 2.0",
      "description": "The objectives of the standard are to provide a common set
of terminology and definitions for the documentation of digital geospatial data. The
```

standard establishes the names of data elements and compound elements (groups of data elements) to be used for these purposes, the definitions of these compound elements and data elements, and information about the values that are to be provided for the data elements.",

```
    "created": "2016-10-10T16:10:01.213Z",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/mediation/schemas/643d2431ff31955de485f84cbbc598d1"
      }
    }
  }, {
    "id": "66dcd67e36e93540bc77e4710fd749ad",
    "uri": "http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2",
    "type": "schema:Schema",
    "title": "National System for Geospatial Intelligence Metadata
Implementation Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2.",
    "description": "An XML encoding of the NMIS logical model (Part 1) that is
conformant to ISO 19139 and the XML Data Encoding Specification for Information
Security Marking Metadata (DES.ISM.XML.V9)",
    "created": "2016-10-10T16:09:58.704Z",
    "namespace": "http://metadata.ces.mil/dse/ns/GSIP/5.0/nas",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/mediation/schemas/66dcd67e36e93540bc77e4710fd749ad"
      }
    }
  }, {
    "id": "3712ab70e045d43a416ce468d295e0ea",
    "uri": "http://www.isotc211.org/2005/gmd",
    "type": "schema:Schema",
    "title": "ISO 19139 Schema (version 2012-07-13)",
    "description": "Geographic MetaData (GMD) extensible markup language is a
component of the XML Schema Implementation of Geographic Information Metadata
documented in ISO/TS 19139:2007. GMD includes all the definitions of
http://www.isotc211.org/2005/gmd namespace. The root document of this namespace is the
file gmd.xsd.",
    "created": "2016-10-10T16:09:56.115Z",
    "namespace": "http://www.isotc211.org/2005/gmd",
    "_links": {
      "self": {
        "href":
"http://54.208.90.94/mediation/schemas/3712ab70e045d43a416ce468d295e0ea"
      }
    }
  }, {
    "id": "32ed1ba2e79e51ed951b250ea1bd50d9",
    "uri": "http://metadata.dod.mil/mdr/ns/DDMS/2.0",
    "type": "schema:Schema",
    "title": "Department of Defense Discovery Metadata Specification (DDMS),
```

```

Version 2.0",
  "description": "Defines discovery metadata elements for resources posted
to community and organizational shared spaces",
  "created": "2016-10-10T16:10:04.174Z",
  "namespace": "http://metadata.dod.mil/mdr/ns/DDMS/2.0/",
  "_links": {
    "self": {
      "href":
"http://54.208.90.94/mediation/schemas/32ed1ba2e79e51ed951b250ea1bd50d9"
    }
  }
}, {
  "id": "f981d111fdbbf8e7916f6dfe9dcc7c69",
  "uri": "urn:uuid:566ff231-e768-41de-bb26-e0def91ceb9b",
  "type": "schema:Schema",
  "title": "National System for Geospatial Intelligence Metadata
Implementation Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2.0",
  "description": "An XML encoding of the NMIS logical model (Part 1) that is
conformant to ISO 19139 and the XML Data Encoding Specification for Information
Security Marking Metadata (DES.ISM.XML.V9).",
  "created": "2016-11-04T14:17:04.903Z",
  "namespace": "http://metadata.ces.mil/dse/ns/GSIP/5.0/nas",
  "_links": {
    "self": {
      "href":
"http://54.208.90.94/mediation/schemas/f981d111fdbbf8e7916f6dfe9dcc7c69"
    }
  }
}, {
  "id": "315a1576cd09615683160f9f1e84ca97",
  "uri": "urn:uuid:02200156-4271-4d52-a2e1-47d268680aa7",
  "type": "schema:Schema",
  "title": "Department of Defense Discovery Metadata Specification (DDMS),
Version 2.0",
  "description": "Defines discovery metadata elements for resources posted
to community and organizational shared spaces",
  "created": "2016-11-04T14:17:05.619Z",
  "namespace": "http://metadata.dod.mil/mdr/ns/DDMS/2.0/",
  "_links": {
    "self": {
      "href":
"http://54.208.90.94/mediation/schemas/315a1576cd09615683160f9f1e84ca97"
    }
  }
}]
},
"_links": {
  "self": {
    "href": "http://54.208.90.94/mediation/schemas"
  }
},
"service": {

```



```

    "href": "http://54.208.90.94/mediation"
  },
  "curies": [{
    "href": "http://www.opengis.net/rels/mediation/{rel}",
    "name": "mediation",
    "templated": true
  }]
},
"aggregations": [],
"page": {
  "size": 20,
  "totalElements": 6,
  "totalPages": 1,
  "number": 0
}
}

```

## Schema

The Schema resource is used to retrieve, update, and delete schema instance.

### Retrieve a schema

To retrieve a particular instance of schema, a HTTP **GET** request will get the details of a schema. There are two ways to retrieve an instance of a schema, using an internal id in the path or by using its Linked Data URL (using uri as query parameter).

### Query Parameters

None

### Response structure

The response of the schema instance request is conform to the [JSON Schema Model](#).

### Links

The following table describes the link relation types accessible from a schema instance that provide transitions to other states related to the schema instance, in particular the relevant schema mappings that use this schema instance.

Relation	Description
<b>self</b>	Refers to this <a href="#">schema</a> itself
<b>mediation:schemas</b>	Refers to schemas search endpoint
<b>mediation:sourceSchemaMappings</b>	Refers to schema mappings that accepts this schema as a source schema
<b>mediation:targetSchemaMappings</b>	Refers to schema mappings that accepts this schema as a target schema

Relation	Description
<code>mediation:schemaMappings</code>	Refers to schema mappings having this schema either as a source or as target schema
<code>mediation:validator</code>	Refers to validator service for this schema
<code>service</code>	Refers to the mediation service
<code>curies</code>	Refers to the curies defined for the links

### Examples

===== HAL+JSON Format Response (Level 3 REST API)

The following HTTP request performs a GET Request to get an instance of a schema in HAL+JSON format.

```
GET /schemas/66dcd67e36e93540bc77e4710fd749ad HTTP/1.1
Accept: application/hal+json
Host: localhost
```

The HAL+JSON response contains the description of the schema in JSON-LD (which can be converted to Linked Data by applying the JSON-LD context of the service). In addition, it provides links to other states than be followed by clients following the semantic of the link relation types described in the previous section.

```
{
  "id": "66dcd67e36e93540bc77e4710fd749ad",
  "uri": "http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2",
  "type": "schema:Schema",
  "register": [
    "schemas"
  ],
  "title": "National System for Geospatial Intelligence Metadata Implementation Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2.",
  "description": "An XML encoding of the NMIS logical model (Part 1) that is conformant to ISO 19139 and the XML Data Encoding Specification for Information Security Marking Metadata (DES.ISM.XML.V9)",
  "created": "2016-10-10T16:09:58.704Z",
  "publisher": [{
    "name": "National Geospatial Intelligence Agency (NGA)",
    "uri": "http://www.nga.mil",
    "type": "org:Organization"
  }],
  "keyword": [
    "ISO 19139 Profile",
    "XML Schema",
    "Metadata",
    "NGA",
  ]
}
```

```

    "Geospatial",
    "NMIS"
  ],
  "namespace": "http://metadata.ces.mil/dse/ns/GSIP/5.0/nas",
  "schemaLanguage": "http://www.w3.org/TR/xmlschema-1/",
  "distribution": [{
    "title": "NGA.STND.0018_2.2 XML Schema",
    "description": "NGA.STND.0018_2.2 XML schema encoding",
    "accessURL": "http://schemas.opengis.net/iso/19139/20070417/gmd/gmd.xsd",
    "type": "dcat:Distribution",
    "representationTechnique": "http://www.w3.org/TR/xmlschema-1/"
  }],
  "alternative": "NGA.STND.0018_2.2",
  "_links": {
    "self": {
      "href":
"http://54.208.90.94/mediation/schemas/66dcd67e36e93540bc77e4710fd749ad"
    },
    "service": {
      "href": "http://54.208.90.94/mediation"
    },
    "mediation:validator": {
      "href":
"http://54.208.90.94/mediation/schemas/66dcd67e36e93540bc77e4710fd749ad/validator"
    },
    "mediation:schemas": {
      "href": "http://54.208.90.94/mediation/schemas"
    },
    "mediation:sourceSchemaMappings": {
      "href":
"http://54.208.90.94/mediation/schemaMappings?constraint=sourceSchema='http%3A%2F%2Fwww.opengis.net%2Ftestbed12%2Fschemas%23NGA.STND.0018_2.2'"
    },
    "mediation:targetSchemaMappings": {
      "href":
"http://54.208.90.94/mediation/schemaMappings?constraint=targetSchema='http%3A%2F%2Fwww.opengis.net%2Ftestbed12%2Fschemas%23NGA.STND.0018_2.2'"
    },
    "mediation:schemaMappings": {
      "href":
"http://54.208.90.94/mediation/schemaMappings?constraint=sourceSchema='http%3A%2F%2Fwww.opengis.net%2Ftestbed12%2Fschemas%23NGA.STND.0018_2.2' OR targetSchema='http%3A%2F%2Fwww.opengis.net%2Ftestbed12%2Fschemas%23NGA.STND.0018_2.2'"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/mediation/{rel}",
      "name": "mediation",
      "templated": true
    }
  ]
}
}

```

## ==== JSON(-LD) Format Response (Level 2 REST API)

The following HTTP request performs a GET Request to get an instance of a schema in HAL+JSON format.

```
GET /schemas/66dcd67e36e93540bc77e4710fd749ad HTTP/1.1
Accept: application/json
Host: localhost
```

The JSON(-LD) response is identical to the HAL+JSON except it does not have the hyperlinks to other states. The client will need to build the URL to reach other states (by reading documentation of API).

```

{
  "id": "66dcd67e36e93540bc77e4710fd749ad",
  "uri": "http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2",
  "type": "schema:Schema",
  "register": [
    "schemas"
  ],
  "title": "National System for Geospatial Intelligence Metadata Implementation Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2.",
  "description": "An XML encoding of the NMIS logical model (Part 1) that is conformant to ISO 19139 and the XML Data Encoding Specification for Information Security Marking Metadata (DES.ISM.XML.V9)",
  "created": "2016-10-10T16:09:58.704Z",
  "publisher": [{
    "name": "National Geospatial Intelligence Agency (NGA)",
    "uri": "http://www.nga.mil",
    "type": "org:Organization"
  }],
  "keyword": [
    "ISO 19139 Profile",
    "XML Schema",
    "Metadata",
    "NGA",
    "Geospatial",
    "NMIS"
  ],
  "namespace": "http://metadata.ces.mil/dse/ns/GSIP/5.0/nas",
  "schemaLanguage": "http://www.w3.org/TR/xmlschema-1/",
  "distribution": [{
    "title": "NGA.STND.0018_2.2 XML Schema",
    "description": "NGA.STND.0018_2.2 XML schema encoding",
    "accessURL": "http://schemas.opengis.net/iso/19139/20070417/gmd/gmd.xsd",
    "type": "dcat:Distribution",
    "representationTechnique": "http://www.w3.org/TR/xmlschema-1/"
  }],
  "alternative": "NGA.STND.0018_2.2",
}

```

===== Turtle Format Response (Linked Data API)

The following HTTP request performs a GET Request to get an instance of a schema in Turtle format.

```

GET /schemas/66dcd67e36e93540bc77e4710fd749ad HTTP/1.1
Accept: text/turtle
Host: localhost

```

```

@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .
@prefix adms: <http://www.w3.org/TR/vocab-adms/#> .
@prefix org: <http://www.socialml.org/ontologies/organization#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix schema-org: <http://schema.org/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

<http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2>
  a schema:Schema ;
  dct:alternative "NGA.STND.0018_2.2" ;
  dct:created "2016-10-10T16:09:58.704Z"^^xsd:dateTime ;
  dct:description "An XML encoding of the NMIS logical model (Part 1)
that is conformant to ISO 19139 and the XML Data Encoding Specification for
Information Security Marking Metadata (DES.ISM.XML.V9)" ;
  dct:publisher <http://www.nga.mil> ;
  dct:title "National System for Geospatial Intelligence Metadata
Implementation Specification (NMIS) - Part 2: XML Exchange Schema, Version 2.2." ;
  schema:namespace <http://metadata.ces.mil/dse/ns/GSIP/5.0/nas> ;
  schema:schemaLanguage <http://www.w3.org/TR/xmlschema-1/> ;
  dcat:distribution [ a dcat:Distribution ;
dct:description "NGA.STND.0018_2.2 XML
schema encoding" ;
dct:title "NGA.STND.0018_2.2 XML
Schema" ;
adms:representationTechnique
<http://www.w3.org/TR/xmlschema-1/> ;
dcat:accessURL
<http://schemas.opengis.net/iso/19139/20070417/gmd/gmd.xsd>
] ;
  dcat:keyword "ISO 19139 Profile" , "XML Schema" , "Metadata" , "NGA"
, "Geospatial" , "NMIS" .

<http://www.nga.mil>
  a org:Organization ;
  foaf:name "National Geospatial Intelligence Agency (NGA)" .

```

## Schema Mapping Collection

A **Schema Mapping** is defining a mapping between a source and target schema. The **Schema Mappings resource** is used to create and search schema mappings.

## JSON Schema

As the registry service can support different item classes and application profiles, the JSON schema supported by the service is not unique. However there is a set of core properties that will always be present for any items in the registry. The following table describes the core minimal properties for a item.

Path	Type	Description	Card.
id	String	Internal identifier for the schema mapping	1
uri	String	Linked Data URI for the schema mapping (equivalent to @id in JSON-LD)	0..1
type	String	The type (class) of the mapping (schema:SchemaMapping or subclass).	1
title	String	The title of the item	1
titleMap	Object	The title map for each language of the title . Each key corresponds to the two letter language identifier name defined in the capabilities ( for example "en")	0..1
description	String	The description of the item	0..1
descriptionMap	Object	The description map for each language of the description. Each key corresponds to the two letter language identifier name defined in the capabilities ( for example "en")	0..1
created	String	The date of creation (XSD datetime format) (generated by the service)	1
modified	String	The date of the last modification (XSD datetime format) (generated by the service)	0..1
sourceSchema	URI	This property stores the schema being mapped <i>from</i> .	1
targetSchema	URI	This property holds the schema being mapped <i>to</i> .	1

Path	Type	Description	Card.
mappingLanguage	URI	This property stores the language used to create a schema mapping.	1
distribution[]	Array	Distribution	0..1
distribution[].type	String	type of the distribution set to dcat:Distribution	0..1
distribution[].accessURL	URL	access url of the distribution	1
distribution[].format	String	format distribution	0..1
distribution[].representationTechnique	URL	representation technique of the distribution content (XSL, XML Schema)	0..1

## Search Schema Mappings

The search of the schema mappings is performed by performing a HTTP **GET** request on the Schema Mappings resource. The schema mapping search supports free text, by ids, by uris, CQL constraint and can return results with aggregations on specific facets.

### Query Parameters

The following query parameters are supported in the query:

The current parameters supported by the GET request are for paging and sorting.

The following query parameters are supported in the query:

Parameter	Description
<b>q</b>	Text to search in textual fields
<b>uri</b>	One or more URIs of schema mapping instances
<b>id</b>	One or more id of schema mapping instances
<b>includeFacet</b>	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.
<b>facet.fieldname</b>	Constraint values of a given facet field name
<b>constraint</b>	A constraint expressed in CQL. This is used to express more advanced query filtering



Parameter	Description
fields	One or more fields to be included in the response. Use JSON path dot notation for referring paths (ex: publisher.name)
pageNumber	The number of the current page as defined in <a href="#">Paging section</a>
pageSize	The count of items on the current page as defined in <a href="#">Paging section</a>
sort	The sorting parameters as defined in <a href="#">Sorting section</a> .

### Example request

The following performs a schema mapping search between a source schema and target

TODO: include::includes/snippets/mediation/list-schema-mapping-example/http-request.adoc[]

```
schemaMappings?constraint=sourceSchema=%27https%3A%2F%2Fwww.fgdc.gov%2Fschemas%2Fmetadata%2Ffgdc-std-001-1998%27%20R%20targetSchema=%27https%3A%2F%2Fwww.fgdc.gov%2Fschemas%2Fmetadata%2Ffgdc-std-001-1998%27
```

### Response structure

The response is structured according the [Schema Mapping JSON Schema](#).

### Links

The following link relation types are provided in the response to allow the transition to others states from the schema mappings embedded in the response.

TODO: include::includes/snippets/mediation/list-schema-mappings-example/links.adoc[]

### Example response

The embedded objects in the response of the request conforms to the `SchemaMapping` definition in the SRIM Schema Application Profile. The rest of the response return paging information and links to other states.

TODO: include::includes/snippets/mediation/list-schema-mappings-example/http-response.adoc[]

## Schema Mapping

The Schema Mapping resource is used to retrieve, update, and delete schema mapping instance. A [Schema Mapping](#) is defined as a transformation from source schema to another. This resource enable search and discovery of schema mappings.

## Retrieve a schema mapping

To retrieve a particular instance of schema mapping, a HTTP **GET** request will get the details of a schema mapping. There are two ways to retrieve an instance of a schema, using an internal id in the path or by using its Linked Data URL (using `uri` as query parameter).

### Query Parameters

None

### Example request

The following HTTP request performs a GET Request to get an instance of a schema mapping:

```
GET /schemaMappings/2ddf2633f660b1d6f281a989faeec3e6 HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response of the schema mapping instance request is defined by the SchemMapping definition defined in the SRIM Schema Application Profile.

### Links

The following table defines the link relation types accessible from a schema mapping instance that provide transitions to other states related to the schema mapping instance.

Relation	Description
<code>self</code>	Refers to this <a href="#">schema</a> itself
<code>mediation:schemas</code>	Refers to schemas search endpoint
<code>mediation:sourceSchema</code>	Refers to source schema accepted by this schema mapping
<code>mediation:targetSchema</code>	Refers to target schema accepted by this schema mapping
<code>mediation:schemaMappings</code>	Refers to schema mappings search endpoint
<code>mediation:transform</code>	Refers to transformation service for this schema mapping that converts a document compliant to the source schema to a document in the target schema of this mapping.
<code>service</code>	Refers to the mediation service
<code>curies</code>	Refers to the curies defined for the links

### Example response

===== HAL+JSON Format Response (Level 3 REST API)

The following HTTP request performs a GET Request to get an instance of a schema mapping converting NMIS to DDMS format in HAL+JSON format. The HAL+JSON response contains the description of the schema mapping in JSON-LD (which can be converted to Linked Data by applying the JSON-LD context of the service). In addition, it provides links to other states than be followed by clients following the semantic of the link relation types described in the previous section.

```
{
  "id": "2ddf2633f660b1d6f281a989faeec3e6",
  "uri": "http://www.opengis.net/testbed12/schemas#MNIS2DDMS",
  "type": "schema:SchemaMapping",
  "register": [
    "schemas"
  ],
  "title": "NMIS 2.2 to DDMS SchemaMapping",
  "description": "Schema Mapping from NMIS 2.2 to DDMS",
  "created": "2016-10-10T16:10:06.490Z",
  "publisher": [{
    "name": "National Geospatial Intelligence Agency (NGA)",
    "uri": "http://www.nga.mil",
    "type": "org:Organization"
  }],
  "keyword": [
    "NAS",
    "DDMS",
    "XSLT",
    "Geospatial",
    "NMIS"
  ],
  "sourceSchema": "http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2",
  "targetSchema": "http://metadata.dod.mil/mdr/ns/DDMS/2.0",
  "mappingLanguage": "https://www.w3.org/TR/xslt20/",
  "distribution": [{
    "title": "NMIS to DDMS SchemaMapping XSLT Distribution",
    "description": "NGA.STND.0018_2.2 to DDMS SchemaMapping XSLT Distribution",
    "accessURL":
"http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&service=CSW-
ebRIM&id=urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4",
    "type": "dcat:Distribution",
    "representationTechnique": "https://www.w3.org/TR/xslt20/"
  }],
  "_links": {
    "self": {
      "href":
"http://54.208.90.94/mediation/schemaMappings/2ddf2633f660b1d6f281a989faeec3e6"
    },
    "service": {
      "href": "http://54.208.90.94/mediation"
    },
    "mediation:schemaMappings": {
      "href": "http://54.208.90.94/mediation/schemaMappings"
    }
  }
}
```

```

    },
    "mediation:transform": {
      "href":
"http://54.208.90.94/mediation/schemaMappings/2ddf2633f660b1d6f281a989faeec3e6/transfo
rm"
    },
    "mediation:sourceSchema": {
      "href":
"http://54.208.90.94/mediation/schemas/instance?uri=http%3A%2F%2Fwww.opengis.net%2Ftes
tbed12%2Fschemas%23NGA.STND.0018_2.2"
    },
    "mediation:targetSchema": {
      "href":
"http://54.208.90.94/mediation/schemas/instance?uri=http%3A%2F%2Fmetadata.dod.mil%2Fmd
r%2Fns%2FDDMS%2F2.0"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/mediation/{rel}",
      "name": "mediation",
      "templated": true
    }
  ]
}
}

```

===== JSON(-LD) Format Response (Level 2 REST API)

The following HTTP request performs a GET Request to get an instance of a schema in HAL+JSON format.

```

GET /schemaMappings/2ddf2633f660b1d6f281a989faeec3e6 HTTP/1.1
Accept: application/json
Host: localhost

```

The JSON(-LD) response is identical to the HAL+JSON except it does not have the hyperlinks to other states. The client will need to build the URL to reach other states (by reading documentation of API).

```

{
  "id": "2ddf2633f660b1d6f281a989faec3e6",
  "uri": "http://www.opengis.net/testbed12/schemas#MNIS2DDMS",
  "type": "schema:SchemaMapping",
  "register": [
    "schemas"
  ],
  "title": "NMIS 2.2 to DDMS SchemaMapping",
  "description": "Schema Mapping from NMIS 2.2 to DDMS",
  "created": "2016-10-10T16:10:06.490Z",
  "publisher": [{
    "name": "National Geospatial Intelligence Agency (NGA)",
    "uri": "http://www.nga.mil",
    "type": "org:Organization"
  }],
  "keyword": [
    "NAS",
    "DDMS",
    "XSLT",
    "Geospatial",
    "NMIS"
  ],
  "sourceSchema": "http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2",
  "targetSchema": "http://metadata.dod.mil/mdr/ns/DDMS/2.0",
  "mappingLanguage": "https://www.w3.org/TR/xslt20/",
  "distribution": [{
    "title": "NMIS to DDMS SchemaMapping XSLT Distribution",
    "description": "NGA.STND.0018_2.2 to DDMS SchemaMapping XSLT Distribution",
    "accessURL":
"http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&service=CSW-
ebRIM&id=urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4",
    "type": "dcat:Distribution",
    "representationTechnique": "https://www.w3.org/TR/xslt20/"
  }]
}

```

===== Turtle Format Response (Linked Data API)

The following HTTP request performs a GET Request to get an instance of a schema in Turtle format.

```

GET /schemaMappings/2ddf2633f660b1d6f281a989faec3e6 HTTP/1.1
Accept: text/turtle
Host: localhost

```

The response in the following:

HTTP/1.1 200 OK  
Content-Type: text/turtle  
Content-Length: 1617

@prefix schema: <http://www.opengis.net/ont/testbed12/srim/profile/schema#> .  
@prefix adms: <http://www.w3.org/TR/vocab-adms/#> .  
@prefix pav: <http://purl.org/pav/> .  
@prefix org: <http://www.socialml.org/ontologies/organization#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .  
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@prefix dct: <http://purl.org/dc/terms/> .  
@prefix dcat: <http://www.w3.org/ns/dcat#> .  
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

```
<http://www.opengis.net/testbed12/schemas#MNIS2DDMS>
  a          schema:SchemaMapping ;
  dct:created      "2016-11-16T15:06:18.246-05:00"^^xsd:dateTime ;
  dct:description  "Schema Mapping from NMIS 2.2 to DDMS" ;
  dct:modified     "2016-11-16T15:06:22.269-05:00"^^xsd:dateTime ;
  dct:publisher    <http://www.nga.mil> ;
  dct:title        "NMIS 2.2 to DDMS SchemaMapping" ;
  schema:mappingLanguage schema:XSLT ;
  schema:sourceSchema
<http://www.opengis.net/testbed12/schemas#NGA.STND.0018_2.2> ;
  schema:targetSchema <http://metadata.dod.mil/mdr/ns/DDMS/2.0> ;
  dcat:distribution   [ a          dcat:Distribution ;
                       dct:description  "NGA.STND.0018_2.2 to
DDMS SchemaMapping XSLT Distribution" ;
                       dct:title        "NMIS to DDMS
SchemaMapping XSLT Distribution" ;
                       adms:representationTechnique schema:XSLT ;
                       dcat:accessURL
<http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&service=CSW-
ebRIM&id=urn:uuid:38ffef44-e1ee-4856-8812-d8b82da497a4>
                       ] ;
  dcat:keyword        "NAS" , "DDMS" , "XSLT" , "Geospatial" , "NMIS" .

<http://www.nga.mil>
  a          org:Organization ;
  foaf:name   "National Geospatial Intelligence Agency (NGA)" .
```

## Validator

A Validator Resource performs validation of data according the schema the validator is associated with.

## Validate data

To validate a particular instance of data against a schema, a HTTP **GET** request can be used to perform validation against a remote resource accessible from a URL or using a POST with the dataset attached.

### Query Parameters

The following query parameters are supported in the query by the GET request.

Parameter	Description	Card.
<code>url</code>	URL of the document to validate against a given schema	1
<code>schema</code>	URL or the id of the schema to use for validation	1

### Links

The following table describes the current link relation types in the HAL validator response

Relation	Description
<code>mediation:schema</code>	Refers to the schema used for validation
<code>mediation:source</code>	Refers to the document that was used for validation
<code>mediation:validator</code>	Refers to validator service
<code>service</code>	Refers to the mediation service
<code>curies</code>	Refers to the curies defined for the links

### Example request

The following performs a GET Request to get the validation of a ISO 19139 document against the schema ISO 19139 XML Schema identified by the id "3712ab70e045d43a416ce468d295e0ea". The response is request in HAL+JSON.

```
GET /validate?url=https%3A%2F%2Fcatalog.data.gov%2Fharvest%2Fobject%2F7a7fccccf-b4ba-469d-a3e8-5489105f0ef0&schema=3712ab70e045d43a416ce468d295e0ea
Accept: application/hal+json
Host: localhost
```

The following performs a POST Request with the dataset to `validate` with the schema associated with the validator.

TODO: include::includes/snippets/mediation/validator-get-example/http-request.adoc[]

## Response structure

If the validation is performed successfully, the service will return a HTTP Code 200 OK with a validation report, adhering the schema described in the table below. If the URL or schema is invalid it will return a error code 400 (bad request).

Path	Type	Description	Cardinality
status	String	Status of the validation ('valid' or 'invalid')	1
schemaId	String	Unique identifier the schema used for validation	1
sourceURL	String	The url of the document that is validated	1
messages[]	Object	Array list of validation messages	0..1
messages[].level	String	The level of the message ('error', 'warning')	1
messages[].description	String	The description of the message	1
messages[].line	int	The line in which the error occurs	0..1
messages[].column	int	The column in which the error occurs	0..1

## Example responses

===== HAL Response

The following snippet shows an example of report for a valid document in HAL+JSON format. The response contains links to the document URL (whether external or stored by service) and the schema description URL.



```

{
  "status": "valid",
  "schemaId": "3712ab70e045d43a416ce468d295e0ea",
  "sourceURL": "https://catalog.data.gov/harvest/object/7a7fccccf-b4ba-469d-a3e8-5489105f0ef0",
  "_links": {
    "mediation:schema": {
      "href": "http://localhost:8083/schemas/3712ab70e045d43a416ce468d295e0ea"
    },
    "mediation:validator": {
      "href": "http://localhost:8083"
    },
    "service": {
      "href": "http://localhost:8083"
    },
    "mediation:source": {
      "href": "https://catalog.data.gov/harvest/object/7a7fccccf-b4ba-469d-a3e8-5489105f0ef0"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/mediation/{rel}",
      "name": "mediation",
      "templated": true
    }]
  }
}

```

The following snippet shows an example of report for an invalid document in HAL+JSON format.

```

{
  "status": "invalid",
  "schemaId": "66dcd67e36e93540bc77e4710fd749ad",
  "sourceURL": "http://54.208.90.94/documents/xml/nmisExampleInstanceFull_ISM6.xml",
  "messages": [{
    "level": "error",
    "description": "cvc-elt.1: Cannot find the declaration of element
'ns:MD_Metadata'."
  }],
  "_links": {
    "mediation:schema": {
      "href": "http://localhost:8083/schemas/66dcd67e36e93540bc77e4710fd749ad"
    },
    "mediation:validator": {
      "href": "http://localhost:8083"
    },
    "service": {
      "href": "http://localhost:8083"
    },
    "mediation:source": {
      "href":
"http://54.208.90.94/documents/xml/nmisExampleInstanceFull_ISM6.xml"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/mediation/{rel}",
      "name": "mediation",
      "templated": true
    }]
  }
}

```

===== JSON Response

The following snippet shows an example of report for a valid document in JSON format.

```

{
  "status": "valid",
  "schemaId": "3712ab70e045d43a416ce468d295e0ea",
  "sourceURL": "https://catalog.data.gov/harvest/object/7a7fccccf-b4ba-469d-a3e8-5489105f0ef0"
}

```

The following snippet shows an example of report for an invalid document in JSON format.

```

{
  "status": "invalid",
  "schemaId": "66dcd67e36e93540bc77e4710fd749ad",
  "sourceURL": "http://54.208.90.94/documents/xml/nmisExampleInstanceFull_ISM6.xml",
  "messages": [{
    "level": "error",
    "description": "cvc-elt.1: Cannot find the declaration of element
'nas:MD_Metadata'."
  }]
}

```

## Transformer Resource

A Transformer Resource performs the transformation data accessible from a URL or from the body of a POST message from a source schema to a target schema.

### Transform data from one schema to another

#### Query Parameters

The following query parameters are supported in the query by the GET request.

Parameter	Description	Card.
<code>url</code>	URL of the document compliant to the source schema parameter that needs to be transformed to the target schema indicated in the targetSchema parameter.	1
<code>sourceSchema</code>	URL or Id of the schema of the source document	1
<code>targetSchema</code>	Url of id of the target schema of the transformation.	1

#### Response structure

If the transformation is performed successfully, we service will return a HTTP Code 200 OK, along with the result dataset compliant with the target schema. If not, a error message will be returned with the code 400 (Bad Request) if the source document is not compliant to the schema, or source and target schema or a path between them cannot be found.

# Appendix F: Semantic Portrayal Service REST API

## Overview

This document describes the initial version of RESTful Semantic Portrayal Service API. We anticipate that a variety of clients may be using the Semantic Portrayal Service, and as a consequence it is difficult to accommodate the needs of every type of client. It is almost certain that the REST API will evolve and be modified as more requirements and features are added to the service. The hypermedia-driven API provides a robust approach to evolve the API without breaking the client ecosystem, as long as the clients are using the semantics of the link relation types. For this reason, the service has adopted a hypermedia-driven RESTful API by default, which meets level 3 of the [Richardson Maturity Model](#).

The default serialization of the information model in the service is JSON, as it is understood by most programming languages. However, to support machine-processable information, we enforce the JSON to be compatible Linked data by using JSON-LD Context. In this way, data can be converted to Linked Data Representation and be reasoned on and linked to other information expressed as Linked Data. The REST API does support content negotiation to return a response in Linked Data Format (RDF/XML, Turtle, NTriple) when applicable. The REST API in this document provides the core minimum functionalities that are based on the **Portrayal Ontologies** described in Appendix C.

### NOTE

The example URL used in this documentation used the hostname <http://localhost:port>. This hostname should be replaced by the entry point (baseURL) of the service that you want to access in the format: <http://{hostname}:{port}/{rootPath}>

## HTTP verbs

The RESTful API tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs. The following HTTP verbs are used by the Service.

Verb	Usage
GET	Used to retrieve a resource
POST	Used to create a new resource
PUT	Used to update an existing resource with a complete update
DELETE	Used to delete an existing resource

## HTTP status codes

The Service REST API tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP status codes. The following table summarizes the status codes and usage.

Status code	Usage
200 OK	The request completed successfully
201 Created	A new resource has been created successfully. The resource's URI is available from the response's <code>Location</code> header
204 No Content	An update to an existing resource has been applied successfully
400 Bad Request	The request was malformed. The response body will include an error providing further information
404 Not Found	The requested resource did not exist
405 Method not allowed	The request was made of a resource using a request method not supported by that resource; for example, using GET on a form which requires data to be presented via POST, or using PUT on a read-only resource.
409 Conflict	The request could not be completed due to a conflict with the current state of the resource. This code is only allowed in situations where it is expected that the user might be able to resolve the conflict and resubmit the request.
415 Unsupported media type	The server is refusing to service the request because the entity of the request is in a format not supported by the requested resource for the requested method.
500 Internal Server Error	The Web server encountered an unexpected condition that prevented it from fulfilling the request by the client

## Headers

Every response has the following header(s):

Name	Description
Content-Type	The Content-Type of the payload, e.g. <code>application/hal+json</code>

Future headers may be added for managing access control to the resources managed by the service.

## Errors

Whenever an error response (status code  $\geq 400$ ) is returned, the body will contain a JSON object that describes the problem. The error object has the following structure:

Path	Type	Description
timestamp	Number	The time, in milliseconds, at which the error occurred
status	Number	The HTTP status code, e.g. <b>400</b>
error	String	The HTTP error that occurred, e.g. <b>Bad Request</b>
message	String	A description of the cause of the error
path	String	The path to which the request was made

For example, a request that attempts to apply a non-existent resource to a symbolset item will produce a **400 Bad Request** response:

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8
Content-Length: 189

{
  "timestamp" : 1461855421813,
  "status" : 400,
  "error" : "Bad Request",
  "message" : "The resource item 'http://localhost:8080/styles/123' does not exist",
  "path" : "/schemas"
}
```

## Paging and Sorting

### Paging

Rather than return everything from a large result set, the REST API recognizes some URL parameters that will influence the page size (**size** parameter) and starting page number (**page** parameter) as well as sorting of the result set (**sort** parameter).

See the following example, where we set the page size to 5 and request the third page (page 2) as page numbers are zero-indexed:

```
GET /symbolsets?page=2&size=5 HTTP/1.1
Host: localhost
```

The paginated results in HAL format returns the following response:

```

{
  "_embedded": {
    ...data...
  },
  "_links": {
    "self": {
      "href": "http://localhost:8080/symbolsets"
    },
    "first": {
      "href": "http://localhost:8080/symbolsets?page=0&size=5"
    },
    "prev": {
      "href": "http://localhost:8080/symbolsets?page=1&size=5"
    },
    "next": {
      "href": "http://localhost:8080/symbolsets?page=3&size=5"
    },
    "last": {
      "href": "http://localhost:8080/symbolsets?page=5&size=5"
    },
  },
  "page": {
    "size": 5,
    "totalElements": 27,
    "totalPages": 6,
    "number": 2
  }
}

```

Each paged response will return links to the first, previous, next, and last page of results based on the current page using the IANA defined link relations [first](#), [prev](#), [next](#), [last](#). If you are currently at the first page of results, however, no [prev](#) link will be rendered. The same is true for the last page of results: no [next](#) link will be rendered.

The paginated results also have extra data about the page settings , including the size of a page, total elements, total pages, and the page number you are currently viewing. This extra information makes it very easy for the consumer to configure UI tools like sliders or indicators to reflect the overall position the user is in viewing the data. For example, the document above shows we are looking at the third page (with page numbers indexed to 0 being the first).

## Sorting

The REST API recognizes sorting parameters. To have your results sorted on a particular property, add a [sort](#) URL parameter with the name of the property you want to sort the results on. You can control the direction of the sort by appending a [,](#) to the the property name plus either [asc](#) or [desc](#).

The following examples will sort results by title in ascending order:

```
GET /symbolsets?sort=title,asc HTTP/1.1
Host: www.mydomain.com
```

To sort the results by more than one property, keep adding as many `sort=PROPERTY` parameters as you need. They will be added in the order they appear in the query string.

## Search Results

A number of endpoints of the service return search results. They usually support Level 2 (JSON) and Level 3 response (HAL+JSON). The search results contain the collection of matched items, paging information and optionally faceted aggregation results. The following describes the response structure for each format.

### HAL+JSON Search Results

Path	Type	Description
<code>_embedded</code>	Array	The HAL <code>_embedded</code> field that contains a collection of instances
<code>embedded_collectionName[]</code>	Array	The array of items instances defined a given JSON schema. The <code>collectionName</code> may vary depending on the types of items contained in the collection.
<code>aggregations[]</code>	Array	The aggregation results as defined in the <a href="#">Aggregation JSON Schema</a> . This field is optional if no faceted search is performed on the endpoint.
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket
<code>_links</code>	Object	Links to other states
<code>page</code>	Object	The page state. See <a href="#">&lt;&lt;_paging,Paging section&gt;</a>



Path	Type	Description
<code>page.size</code>	Number	The page size
<code>page.totalElements</code>	Number	The total elements matched by the search request
<code>page.totalPages</code>	Number	The total number of pages in the results
<code>page.number</code>	Number	The current page number (starts at 0)
<code>_links[]</code>	Array	Array of hypermedia links to other reachable states.

## JSON Search results

When the JSON or JSON-LD response is retrieved, the matched items are placed in an array referred by the *results* field. Aggregations results are present only when faceted search is performed and paging information are returned.

Path	Type	Description
<code>results[]</code>	Array	The array of vocabulary instances that matches the search criteria
<code>aggregations[]</code>	Array	The aggregation results as defined in the <a href="#">Aggregation JSON Schema</a> . This field is optional is no faceted search is performed on the endpoint.
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket
<code>page</code>	Object	The page state. See <<_paging,Paging section>
<code>page.size</code>	Number	The page size

Path	Type	Description
<code>page.totalElements</code>	Number	The total elements matched by the search request
<code>page.totalPages</code>	Number	The total number of pages in the results
<code>page.number</code>	Number	The current page number (starts at 0)

## Aggregation JSON Schema

In many use cases, it is useful to aggregate the search results according some facets values (number of items per topic, per publisher, etc). Each facet is composed of a name, global metrics (right now only total count is supported), a set of buckets containing a label (unique value of the facet) and the total count for each label within the context of the facet and search results. The JSON schema of the Aggregation results is defined in the following table.

Path	Type	Description
<code>aggregations[]</code>	Array	The aggregation results
<code>aggregations[].name</code>	String	The name of the aggregation (or facet)
<code>aggregations[].metrics</code>	Object	The metrics object containing global statistics of the object
<code>aggregations[].metrics.count</code>	Number	The total count of the aggregation
<code>aggregations[].buckets[]</code>	Array	The array of buckets of the aggregation
<code>aggregations[].buckets[].label</code>	String	The label of the bucket
<code>aggregations[].buckets[].count</code>	String	The count of the label for the bucket

## Resources Summary

The following resources are the core resources supported by the Semantic Portrayal Service.

Resources	Description	Operations
Capabilities	This resource describes the capabilities of the service	GET
StyleSet	represents a collection of Styles and supports search on a collection of StyleSet instances based on search criteria	GET
StyleSet	represents an instance of a StyleSet	GET
Styles	represents a collection of Styles and supports search on a collection of Style instances based on search criteria	GET

Resources	Description	Operations
Style	represents an instance of a Style	GET
SymbolSets	represents a collection of Symbol Sets and supports search on a collection of symbol sets based on search criteria	GET
SymbolSet	represents a collection of Symbol Sets and supports search on a collection of symbol sets based on search criteria	GET
SymbolRenderer	This resource supports rendering of a symbol to support legend rendering	GET
LayerRenderer	This resource supports rendering of data for a given style	GET, POST
JSONLD Context	This resource returns the JSON-LD context associated with the JSON representation returned by the service	GET
SPARQL	This resource provides a SPARQL service query endpoint that can accommodate more advanced query capabilities on the portrayal information	GET

## Level 2 REST Endpoints

The following table describes the endpoint URL paths for each resource for a Level 2 REST API. These paths are considered **NON-NORMATIVE** but rather **INFORMATIVE**, as changes of path templates may occur in the future, requiring updates of clients and version control of APIs. The use of hypermedia REST API is advised to be isolated from these changes. However we provide this information to support frameworks that work with Level 2 REST API (such as AngularJS)

Path	HTTP Methods	Consume	Produce
/	GET,HEAD		application/hal+json
/capabilities	GET,HEAD		* application/hal+json * application/json
/stylesets	GET,HEAD		* application/hal+json * application/json

Path	HTTP Methods	Consume	Produce
/stylesets/instance	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/stylesets/{id}	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/styles	GET,HEAD		* application/hal+json * application/json
/styles/instance	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/styles/{id}	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/symbolsets	GET,HEAD		* application/hal+json * application/json

Path	HTTP Methods	Consume	Produce
/symbolsets/instance	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/symbolsets/{id}	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/symbols	GET,HEAD		* application/hal+json * application/json
/symbols/instance	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/symbols/{id}	GET,HEAD		* application/hal+json * application/json * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/symbols/{id}/render	GET,HEAD		* Supported output graphic formats
/styles/{id}/render	GET,HEAD		* Supported output graphic formats

Path	HTTP Methods	Consume	Produce
/styles/{id}/render	POST	* Supported data formats	* Supported output graphic formats
/sparql	GET,HEAD		* application/sparql-results+xml * application/sparql-results+json * text/csv * text/tab-separated-values * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples
/sparql	POST	* application/x-www-form-urlencoded * application/sparql-query	* application/sparql-results+xml * application/sparql-results+json * text/csv * text/tab-separated-values * application/rdf+xml, * text/turtle, * text/n3, * application/n-triples

## Link relation types

The following table describes the list of link relation types used by the Semantic Portrayal Service. We use the OGC namespace to define for the relationships that can be reused by other services such as capabilities, apiDocumentation. The links specific to the portrayal service have the following relation type URI template: <http://www.opengis.net/rels/portrayal/{rel}>.

Relation type	URI	Description
ogc:capabilities	<a href="http://www.opengis.net/rels/capabilities">http://www.opengis.net/rels/capabilities</a>	Reference to the capabilities of the service
portrayal:styles	<a href="http://www.opengis.net/rels/portrayal/styles">http://www.opengis.net/rels/portrayal/styles</a>	Reference to a collection of Style instances in the portrayal service
portrayal:style	<a href="http://www.opengis.net/rels/portrayal/style">http://www.opengis.net/rels/portrayal/style</a>	Reference to an instance of Style in the portrayal service

Relation type	URI	Description
portrayal:symbolSets	<a href="http://www.opengis.net/rels/portrayal/symbolSets">http://www.opengis.net/rels/portrayal/symbolSets</a>	Reference to a collection of SymbolSet instances in the portrayal service
portrayal:symbolSet	<a href="http://www.opengis.net/rels/portrayal/symbolSet">http://www.opengis.net/rels/portrayal/symbolSet</a>	Reference to an instance of SymbolSet in the portrayal service
portrayal:symbols	<a href="http://www.opengis.net/rels/portrayal/symbols">http://www.opengis.net/rels/portrayal/symbols</a>	Reference to a collection of Symbol instances in the portrayal service
portrayal:symbol	<a href="http://www.opengis.net/rels/portrayal/symbol">http://www.opengis.net/rels/portrayal/symbol</a>	Reference to an instance of Symbol in the portrayal service
portrayal:renderer	<a href="http://www.opengis.net/rels/portrayal/renderer">http://www.opengis.net/rels/portrayal/renderer</a>	Reference to the portrayal renderer of a symbol or a map layer
ogc:jsonldContext	<a href="http://www.opengis.net/rels/jsonldContext">http://www.opengis.net/rels/jsonldContext</a>	Reference to the JSON-LD context to apply to the JSON content to transform it to Linked Data
ogc:documentation	<a href="http://www.opengis.net/rels/restdoc">http://www.opengis.net/rels/restdoc</a>	Reference to the REST documentation of the service

## Content negotiation

Most of the resource can serve multiple representations including:

Format	Mime type	Description
HAL+JSON	application/hal+json	It is the <b>default format</b> of the service. The JSON payload is compatible with JSON-LD and is aligned with the Portrayal Ontology Ontology.
JSON-LD	application/ld+json	Compliant with the Portrayal Ontology using the JSON-LD context.
JSON	application/json	Compliant with the Portrayal Ontology Ontology using the JSON-LD context.
RDF/XML	application/rdf+xml	Compliant with the Portrayal Ontology
Turtle	text/turtle	Compliant with the Portrayal Ontology
N-Triples	text/ntriples	Compliant with the Portrayal Ontology

Renderers produces different formats depending of their type such as SVG, KML, PNG etc.

# Semantic Portrayal Resources

This section describes the list of resources made accessible by the Semantic Portrayal Service. The RESTful API has an entry point (service root) which provides links which represents the current state transitions supported by the service.

## Service Root

The root is the entry point of the RESTful Semantic Portrayal service – it is what the client comes into contact with when consuming the API for the first time. If the HATEOAS constraint is to be considered and implemented throughout by clients, then this is the place to start. The root endpoint provides **a set of links** with well-defined relation types that corresponds to the supported capabilities of the service. As the Semantic Portrayal Service API evolves in the future, there would be many more links, each with it's own semantics defined by the type of [link relation](#).

The API is considered as RESTful as it is fully discoverable **from the root** and with **no prior knowledge** – meaning the client should be able to navigate the API by doing a GET on the root. Moving forward, all state changes are driven by the client using the available and discoverable transitions that the REST API provides in representations (hence Representational State Transfer).

## Accessing the root endpoint

To get access to the root of the service and get the list of links, a **GET** request is sent to the base url of the service.

### Request structure

This request sends a 'GET' request to the base url of the service. Here the HTTP request associated to get the list of links supported by the service:

```
GET / HTTP/1.1
Host: localhost
```

### Query Parameters

None

### Response structure

The response is returning a HAL response (application/hal+json) with the **\_links** object that contains links associated with relation types (used as property names in JSON).

Path	Type	Description
@context	String	<a href="#">JSON-LD Context</a> applicable to JSON response of the service
type	String	The type of the resource (always Service)



Path	Type	Description
title	String	Title of the service
description	String	Description of the service
categories	Array	Categories of the service
_links	Object	<a href="#">Links</a> to other resources

## Links

The following table describes the current link relation types supported by the service.

Relation	Description
portrayal:capabilities	Refers to the <a href="#">capabilities</a> of this service.
portrayal:styles	Refers to the <a href="#">style search endpoint</a> supported by this service
portrayal:symbols	Refers to the <a href="#">symbol search endpoint</a> supported by this service
portrayal:symbolSets	Refers to the <a href="#">symbolSet search endpoint</a> supported by this service
portrayal:renderer	Refers to the <a href="#">renderer endpoint</a> supported by this service
portrayal:sparql	Refers to the <a href="#">SPARQL endpoint</a> supported by this service
ogc:jsonLdContext	Refers to JSON Context
curies	The curies to use to expand the link relation types to URI

## Example response

The following example response shows the current links supported by the service.

HTTP/1.1 200 OK

Content-Type: application/hal+json

Content-Length: 921

```
{
  "type": "srim:Service",
  "title": "Semantic Portrayal Service",
  "description": "Semantic Portrayal Service prototype for OGC Testbed12",
  "category": [
    "http://www.opengis.net/specs/testbed12/semanticPortrayalService"
  ],
  "publisher": [{
    "name": "Image Matters LLC",
    "uri": "http://www.imagemattersllc.com",
    "type": "org:Organization"
  }],
  "version": "0.1",
  "_links": {
    "self": {
      "href": "http://localhost:8082"
    },
    "portrayal:capabilities": {
      "href": "http://localhost:8082/capabilities"
    },
    "portrayal:symbols": {
      "href": "http://localhost:8082/symbols"
    },
    "portrayal:symbolsets": {
      "href": "http://localhost:8082/symbolsets"
    },
    "portrayal:styles": {
      "href": "http://localhost:8082/styles"
    },
    "portrayal:renderer": {
      "href": "http://localhost:8082/renderer"
    },
    "ogc:jsonldContext": {
      "href": "http://localhost:8082/context"
    },
    "portrayal:sparql": {
      "href": "http://localhost:8082/sparql"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/portrayal/{rel}",
      "name": "portrayal",
      "templated": true
    }
  ]
}
```

## Capabilities

This resource describes the capabilities of the service, including the supported portrayal items, application profiles, formats.

### Query Parameters

There is no query parameter to retrieve the capabilities.

### Example request

The following is an example of GET Request performed on the Capability resource.

```
GET /capabilities HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response of Capability request has the following structure:

Path	Type	Description
<code>itemClasses[]</code>	Array	ItemClasses supported by the registry
<code>itemClasses[].id</code>	String	The ID of the ItemClass
<code>itemClasses[].uri</code>	String	The URI of the ItemClass
<code>itemClasses[].type</code>	String	The RDF type of the ItemClass (always <code>sr:ItemClass</code> )
<code>itemClasses[].title</code>	String	The human readable title of the ItemClass
<code>itemClasses[].description</code>	String	The human readable description of the ItemClass
<code>formats[]</code>	Array	The supported graphic format mime types
<code>_links</code>	Object	The hypermedia links to other states

### Links

The following table describes the current link relation types in the HAL capabilities response

Relation	Description
<code>service</code>	Reference to the entry point of the service

Relation	Description
<code>self</code>	Refers to this endpoint.
<code>curies</code>	Refers to the curies defined for the links

### Example response

The following example response shows a capability of the service is represented.

```

HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 30456

{
  "formats": [
    "image/svg+xml",
    "image/png",
    "image/jpeg",
    "image/gif",
    "image/geotiff"
  ],
  "itemClasses": [{
    "id": "style:Style",
    "uri": "http://www.opengis.net/ont/portrayal/style#Style",
    "type": "srim:ItemClass",
    "title": "Style",
    "description": "Style class"
  }, {
    "id": "style:StyleSet",
    "uri": "http://www.opengis.net/ont/portrayal/style#StyleSet",
    "type": "srim:ItemClass",
    "title": "StyleSet",
    "description": "StyleSet class"
  }, {
    "id": "style:SymbolSet",
    "uri": "http://www.opengis.net/ont/portrayal/symbol#SymbolSet",
    "type": "srim:ItemClass",
    "title": "SymbolSet",
    "description": "SymbolSet class"
  }, {
    "id": "style:PortrayalRuleSet",
    "uri": "http://www.opengis.net/ont/portrayal/style#PortrayalRuleSet",
    "type": "srim:ItemClass",
    "title": "Portrayal RuleSet",
    "description": "PortrayalRuleSet class"
  }, {
    "id": "style:FeatureTypeStyle",
    "uri": "http://www.opengis.net/ont/portrayal/style#FeatureTypeStyle",
    "type": "srim:ItemClass",
    "title": "FeatureType Style",

```

```

    "description": "FeatureType class"
  }, {
    "id": "symbol:Symbol",
    "uri": "http://www.opengis.net/ont/portrayal/symbol#Symbol",
    "type": "srim:ItemClass",
    "title": "Portrayal Symbol",
    "description": "Symbol class"
  }, {
    "id": "symbol:SymbolSet",
    "uri": "http://www.opengis.net/ont/portrayal/symbol#SymbolSet",
    "type": "srim:ItemClass",
    "title": "Symbol Set",
    "description": "SymbolSet Class"
  }, {
    "id": "style:PortrayalRule",
    "uri": "http://www.opengis.net/ont/portrayal/style#PortrayalRule",
    "type": "srim:ItemClass",
    "title": "Portrayal Rule",
    "description": "Portrayal Rule"
  }, {
    "id": "style:PortrayalRuleList",
    "uri": "http://www.opengis.net/ont/portrayal/style#PortrayalRuleList",
    "type": "srim:ItemClass",
    "title": "Portrayal RuleList",
    "description": "Portrayal RuleList "
  }, {
    "id": "style:PortrayalRuleList",
    "uri": "http://www.opengis.net/ont/portrayal/style#PortrayalRuleList",
    "type": "srim:ItemClass",
    "title": "Portrayal RuleList",
    "description": "Portrayal RuleList "
  }
}],
"_links": {
  "self": {
    "href": "http://localhost:8082/capabilities"
  },
  "service": {
    "href": "http://localhost:8082"
  }
}
}

```

## JSON-LD Context

The JSON produced by the Semantic Portrayal Service is compatible with the Portrayal SRIM Application Profile using the Portrayal ontologies described in Appendix C, by using JSON-LD context. The context is made accessible through an endpoint, so it can be referred and imported by JSON-LD processor to be converted into a Linked Data representation adhering to the Portrayal SRIM Profile.

## Retrieve JSON Context

### Query Parameters

There is no query parameter to retrieve the JSON-LD context.

### Example request

The following is an example of GET Request performed on the JSON-LD context resource.

```
GET /context HTTP/1.1
Host: localhost
```

### Response structure

The response of JSON-LD Context is conforming to the standard JSON-LD.

Path	Type	Description
@context	Object	<a href="#">JSON-LD Context</a> applicable to JSON response of the service

### Example response

The following example response shows a JSON-LD context of the service.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 2222

{
  "@context": {
    "style": "http://www.opengis.net/ont/portrayal/style#",
    "symbol": "http://www.opengis.net/ont/portrayal/symbol#",
    "symbolizer": "http://www.opengis.net/ont/portrayal/symbolizer#",
    "graphic": "http://www.opengis.net/ont/portrayal/graphic#",
    "pav": "http://purl.org/pav/",
    "dct": "http://purl.org/dc/terms/",
    "owl": "http://www.w3.org/2002/07/owl#",
    "xsd": "http://www.w3.org/2001/XMLSchema#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "srim": "http://www.opengis.net/ont/testbed/12/srim#",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "dcat": "http://www.w3.org/ns/dcat#",
    "iso639-2": "http://id.loc.gov/vocabulary/iso639-2/",
    "lingvoj": "http://www.lingvoj.org/ontology#",
    "foaf": "http://xmlns.com/foaf/0.1/",
    "ldp": "http://www.w3.org/ns/ldp#",
    "locn": "http://www.w3.org/ns/locn#",
    "adms": "http://www.w3.org/TR/vocab-adms/#",
```

```

"extent": "http://www.opengis.net/ont/extent#",
"idvoc": "http://www.opengis.net/ont/identifiser#",
"link": "http://www.opengis.net/ont/link#",
"gr": "http://www.heppnetz.de/ontologies/goodrelations/v1#",
"org": "http://www.socialml.org/ontologies/organization#",
"schema": "http://www.opengis.net/ont/testbed12/srim/profile/schema#",
"vcard": "http://www.w3.org/2006/vcard/ns#",
"type": "@type",
"uri": {
  "@id": "rdf:id",
  "@container": "@language"
},
"value": "@value",
"lang": "@language",
"results": "ldp:contains",
"title": "dct:title",
"titleMap": {
  "@id": "dct:title",
  "@container": "@language"
},
"description": "dct:description",
"descriptionMap": {
  "@id": "dct:description",
  "@container": "@language"
},
"prefLabel": "skos:prefLabel",
"prefLabelMap": {
  "@id": "skos:prefLabel",
  "@container": "@language"
},
"category": {
  "@id": "dct:type",
  "@type": "@id"
},
"version": "pav:version",
"creator": {
  "@id": "dct:creator",
  "@type": "@id"
},
"versionNotes": "adms:versionNotes",
"versionNotesMap": {
  "@id": "adms:versionNotes",
  "@container": "@language"
},

"contributor": {
  "@id": "dct:contributor",
  "@type": "@id"
},
"publisher": {
  "@id": "dct:publisher",

```

```

    "@type": "@id"
  },
  "language": {
    "@id": "dct:language",
    "@type": "@id"
  },
  "iso2Code": "lingvoj:iso1",
  "iso3Code": "lingvoj:iso2",
  "label": "rdfs:label",
  "name": "foaf:name",
  "nameMap": {
    "@id": "foaf:name",
    "@container": "@language"
  },
  "sameAs": {
    "@id": "owl:sameAs",
    "@type": "@id"
  },
  "distribution": {
    "@id": "dcat:distribution",
    "@type": "@id"
  },
  "license": {
    "@id": "dct:license",
    "@type": "@id"
  },
  "created": "dct:created",
  "issued": "dct:issued",
  "modified": "dct:modified",

  "mediaType": "dcat:mediaType",
  "country-name": "vcard:country-name",
  "region": "vcard:region",
  "locality": "vcard:locality",
  "postal-code": "vcard:postal-code",
  "street-address": "vcard:street-address",

  "inScheme": {
    "@id": "skos:inScheme",
    "@type": "@id"
  },
  "conformsTo": {
    "@id": "dct:conformsTo",
    "@type": "@id"
  },
  "page": {
    "@id": "foaf:page",
    "@type": "@id"
  },
  "rights": {
    "@id": "dct:rights",

```



```

    "@type": "@id"
  },

  "hasValue": {
    "@id": "vcard:hasValue",
    "@type": "@id"
  },

  "landingPage": {
    "@id": "dcat:landingPage",
    "@type": "@id"
  },

  "accrualPeriodicity": {
    "@id": "dct:accrualPeriodicity",
    "@type": "@id"
  },

  "purpose": "srim:purpose",

  "itemClass": {
    "@id": "srim:itemClass",
    "@type": "@id"
  },

  "keyword": "dcat:keyword",
  "identifier": "dct:identifier",

  "theme": {
    "@id": "dcat:theme",
    "@type": "@id"
  },

  "audience": {
    "@id": "dct:audience",
    "@type": "@id"
  },

  "function": {
    "@id": "srim:function",
    "@type": "@id"
  },

  "subject": {
    "@id": "dct:subject",
    "@type": "@id"
  },

  "project": {
    "@id": "foaf:project",
    "@type": "@id"
  },

  "hasIdentifier": {
    "@id": "id:hasIdentifier",
    "@type": "@id"
  },

  "depiction": {
    "@id": "foaf:depiction",

```

```

    "@type": "@id"
  },
  "provenance": {
    "@id": "dct:provenance",
    "@type": "@id"
  },
  "contactPoint": {
    "@id": "dcat:contactPoint",
    "@type": "@id"
  },
  "hasGeographicExtent": {
    "@id": "extent:hasGeographicExtent",
    "@type": "@id"
  },
  "temporal": {
    "@id": "dct:temporal",
    "@type": "@id"
  },
  "spatial": {
    "@id": "dct:spatial",
    "@type": "@id"
  },
  "accessRights": {
    "@id": "dct:accessrights",
    "@type": "@id"
  },
  "rightsHolder": {
    "@id": "dct:rightsHolder",
    "@type": "@id"
  },
  "subOrganizationOf": {
    "@id": "org:subOrganizationOf",
    "@type": "@id"
  },
  "item": {
    "@id": "srim:item",
    "@type": "@id"
  },
  "status": "srim:status",
  "itemType": {
    "@id": "srim:itemType",
    "@type": "@id"
  },
  "organization-name": "vcard:organization-name",
  "address": {
    "@id": "vcard:address",
    "@type": "@id"
  }

```

```

},
"fn": "vcard:fn",
"hasEmail": {
  "@id": "vcard:hasEmail",
  "@type": "@id"
},
"vcardTitle": "vcard:title",
"hasTelephone": {
  "@id": "vcard:hasTelephone",
  "@type": "@id"
},
"featureType": "style:featureType",
"rules": "style:hasRule",
"symbolSet": "symbol:symbolSet",
"hasRuleSet": "style:hasRuleSet",
"styles": "style:hasStyle",
"ruleset": "style:hasRuleSet",
"minScaleDenominator": "style:minScaleDenominator",
"maxScaleDenominator": "style:maxScaleDenominator",
"constraint": "style:hasConstraint",
"constraintLanguage": "style:constraintLanguage",
"constraintLanguageVersion": "style:constraintLanguageVersion",
"ruleCondition": "style:has",
"body": "style:body",
"browseGraphic": "symbol:browseGraphic",
"specification": "symbol:specification",
"denotes": "symbol:denotes",
"uom": "symbolizer:uom",
"comp-op": "symbolizer:comp-op",
"symbolizer": "symbolizer:symbolizer",
"geometryProperty": "symbolizer:geometryProperty",
"propertyName": "symbolizer:propertyName",
"hasGraphicContent": "graphic:hasGraphicContent",
"graphicObject": "graphic:graphicObject",
"graphicSymbol": "graphic:graphicSymbol",
"mark": "graphic:shape",
"shape": "graphic:graphicObject",
"hasGraphicProperty": "graphic:hasGraphicProperty",
"fontFamily": "graphic:fontFamily",
"fontStyle": "graphic:fontStyle",
"hasColor": "graphic:hasColor",
"fill": "graphic:fill",
"font": "graphic:font",
"graphicStroke": "graphic:graphicStroke",
"graphicFill": "graphic:graphicFill",
"onlineResource": "graphic:onlineResource",
"stroke": "graphic:stroke",
"wellKnownShape": "graphic:wellKnownShape",
"labelPlacement": "graphic:labelPlacement",
"pointPlacement": "graphic:pointPlacement",
"linePlacement": "graphic:linePlacement",

```

```

"graphicContent": "graphic:graphicContent",
"externalGraphic": "graphic:externalGraphic",
"graphicProperty": "graphic:graphicProperty",
"colorName": "graphic:colorName",
"cssColor": "graphic:cssColor",
"fill-property": "graphic:fill-property",
"fill-color": "graphic:fill-color",
"fill-opacity": "graphic:fill-opacity",
"font-property": "graphic:font-property",
"font-code": "graphic:font-code",
"font-size": "graphic:font-size",
"font-weight": "graphic:font-weight",
"format": "graphic:format",
"gap": "graphic:gap",
"halo": "graphic:halo",
"radius": "graphic:radius",
"asWKT": "graphic:asWKT",
"initialGap": "graphic:initialGap",
"perpendicularOffset": "graphic:perpendicularOffset",
"stroke-property": "graphic:stroke-property",
"stroke-color": "graphic:stroke-color",
"stroke-dasharray": "graphic:stroke-dasharray",
"stroke-dashoffset": "graphic:stroke-dashoffset",
"stroke-linecap": "graphic:stroke-linecap",
"stroke-linejoin": "graphic:stroke-linejoin",
"stroke-opacity": "graphic:stroke-opacity",
"stroke-linejoin": "graphic:stroke-linejoin",
"stroke-width": "graphic:stroke-width",
"svgPath": "graphic:svgPath",
"anchorPointX": "graphic:anchorPointX",
"anchorPointY": "graphic:anchorPointY",
"opacity": "graphic:opacity",
"inlineContent": "graphic:inlineContent",
"markIndex": "graphic:markIndex",
"textLabel": "graphic:textLabel",
"isRepeated": "graphic:isRepeated",
"isAligned": "graphic:isAligned",
"generalizeLine": "graphic:generalizeLine";
}
}

```

## Styles

A **Style** defines the set of portrayal rules to apply to some geospatial data. **Style** has subclasses such as `style:FeatureTypeStyle` and **style:CoverageStyle**. The **Styles resource** is used to search style instances based on some search criteria.

## JSON Schema

The following defines the JSON schema for the `Style` derived from the `Style` Ontology defined in

Appendix C. It can also be derived from the JSON-LD Context.

Path	Type	Description	Card.
id	String	Internal identifier for the symbol (which can be used in Level 2 API)	1
uri	String	Linked Data URI for the Style (equivalent to @id in JSON-LD)	0..1
type	String	The type (class) of the Style (symbol:Style subclasses such as style:FeatureTypeStyle).	1
title	String	The title of the Style	1
titleMap	Object	The title map for each language of the title . Each key corresponds to the two letter language identifier name ( for example "en")	0..1
description	String	The description of the Style	0..1
descriptionMap	Object	The description map for each language of the description. Each key corresponds to the two letter language identifier name ( for example "en")	0..1
created	String	The date of creation (XSD datetime format) (generated by the service)	1
modified	String	The date of the last modification (XSD datetime format) (generated by the service)	0..1
featureType	uri	URI of FeatureType associated with a FeatureTypeStyle instance	0..1

Path	Type	Description	Card.
rules[]	Object	Array of PortrayalRule URIs. See ontology for Symbolizer in Appendix C	1

## Search Styles

The search of the Styles is performed by performing a HTTP **GET** request on the Styles resource. The style search supports free text, by types, ids, or uris, CQL constraint and can return results with aggregations on specific facets.

### Query Parameters

The following query parameters are supported in the query:

Parameter	Description	Cardi nality
<b>q</b>	Text to search in textual fields	0..1
<b>type</b>	One or more Style types (style:FeatureTypeStyle or other subclasses)	0..n
<b>uri</b>	One or more URIs of Style instances	0..n
<b>id</b>	One or more id of Style instances	0..n
<b>includeFacet</b>	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.	0..1
<b>facet.fieldname</b>	Constraint values of a given facet field name	0..n
<b>constraint</b>	A constraint expressed in CQL. This is used to express more advanced query filtering	0..1
<b>fields</b>	One or more fields to be included in the response. Use JSON path dot notation for referring paths	0..n
<b>pageNumber</b>	The number of the current page as defined in <a href="#">Paging section</a>	0..1
<b>pageSize</b>	The count of items on the current page as defined in <a href="#">Paging section</a>	0..1
<b>sort</b>	The sorting parameters as defined in <a href="#">Sorting section</a> .	0..n

### Example request

The following request performs a style search of all types

```
GET /styles HTTP/1.1
Accept: application/hal+json
Host: localhost
```

The following request search styles for the featureType 'http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident' using a CQL constraint

```
(featureType='http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident')
```

```
GET
/styles?constraint=(featureType=%27http%3A%2F%2Fwww.opengis.net%2Ftestbed11%2Font%2Fincident%2Fems%23EMSIncident%27) HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response search is structured according the [Search Results Schema](#). The items of the results conforms to the [Style JSON Schema](#). The collection name used in the `_embedded` section of the HAL response is called `portrayal:items`.

### Links

The following link relation types are provided in the response to allow the transition to others states from the Style search results embedded in the response.

Relation	Description
<code>self</code>	Refers to this resource itself
<code>service</code>	Refers to the root of the portrayal service
<code>curies</code>	Refers to the curies defined for the links
<code>first</code>	The first page of results
<code>last</code>	The last page of results
<code>next</code>	The next page of results
<code>prev</code>	The previous page of results

### Example response

===== HAL+JSON Response

The embedded objects in the response of the request conforms to the [Style JSON Schema](#). The collection name used in the `_embedded` section of the HAL response is called `portrayal:items`. The rest of the response return paging information and links to other states.

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 2707

{
```

```

    "_embedded": {
      "portrayal:items": [{
        "id": "345a19896b477721787a82742cc04770",
        "uri":
"http://www.opengis.net/testbed/12/portrayal/ems/style#EMSIncidentStyle",
        "type": "style:FeatureTypeStyle",
        "title": "EMS Incident Type Style",
        "description": "EMS Incident Type Style",
        "created": "2016-11-18T03:18:12.661Z",
        "modified": "2016-11-18T03:18:15.024Z",
        "_links": {
          "self": {
            "href":
"http://localhost:8082/styles/345a19896b477721787a82742cc04770"
          }
        }
      }, {
        "id": "c7a30cb460ad27d98fc4ddd303a7c4ab",
        "uri":
"http://www.opengis.net/testbed/12/portrayal/hswg/style#HSWGIncidentStyle",
        "type": "style:FeatureTypeStyle",
        "title": "HSWG Incident Type Style",
        "description": "HSWG Incident Type Style",
        "created": "2016-11-18T03:18:15.457Z",
        "modified": "2016-11-18T03:18:15.842Z",
        "_links": {
          "self": {
            "href":
"http://localhost:8082/styles/c7a30cb460ad27d98fc4ddd303a7c4ab"
          }
        }
      }
    ],
    "_links": {
      "self": {
        "href": "http://localhost:8082/styles"
      },
      "service": {
        "href": "http://localhost:8082"
      },
      "curies": [{
        "href": "http://www.opengis.net/rels/portrayal/{rel}",
        "name": "portrayal",
        "templated": true
      }
    ]
  },
  "aggregations": [],
  "page": {
    "size": 20,
    "totalElements": 2,
    "totalPages": 1,

```



```
    "number": 0
  }
}
```

===== JSON Response

The result objects in the response of the request conforms to the [Style JSON Schema](#). The rest of the response return paging information.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 850

{
  "results": [{
    "id": "345a19896b477721787a82742cc04770",
    "uri":
"http://www.opengis.net/testbed/12/portrayal/ems/style#EMSIncidentStyle",
    "type": "style:FeatureTypeStyle",
    "title": "EMS Incident Type Style",
    "description": "EMS Incident Type Style",
    "created": "2016-11-18T03:18:12.661Z",
    "modified": "2016-11-18T03:18:15.024Z"
  }, {
    "id": "c7a30cb460ad27d98fc4ddd303a7c4ab",
    "uri":
"http://www.opengis.net/testbed/12/portrayal/hswg/style#HSWGIncidentStyle",
    "type": "style:FeatureTypeStyle",
    "title": "HSWG Incident Type Style",
    "description": "HSWG Incident Type Style",
    "created": "2016-11-18T03:18:15.457Z",
    "modified": "2016-11-18T03:18:15.842Z"
  }],
  "page": {
    "size": 20,
    "number": 0,
    "totalElements": 2
  }
}
```

## Style

The Style resource is used to retrieve a Style instance.

### Retrieve a Style

To retrieve a particular instance of Style, a HTTP **GET** request will get the details of a Style. There are two ways to retrieve an instance of a style, using an internal id in the path or by using its Linked Data URL (using uri as query parameter).

## Query Parameters

Parameter	Description	Cardinality
uri	URI of the style if the instance needs to be retrieved by URI. URI should be encoded to escape special characters.	0..1

### Example request

The following HTTP request performs a GET Request to get an instance of a style identified by its internal id `345a19896b477721787a82742cc04770`

```
GET /styles/345a19896b477721787a82742cc04770 HTTP/1.1
Accept: application/hal+json
Host: localhost
```

The following query get the instance identified by the URI <http://www.opengis.net/testbed/12/portrayal/ems/style#EMSIncidentStyle>

```
GET
/styles/instance?uri=http%3A%2F%2Fwww.opengis.net%2Ftestbed%2F12%2Fportrayal%2Fems%2Fstyle%23EMSIncidentStyle HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response of the instance request can be retrieved in HAL+JSON, JSON-LD and Linked Data formats. The JSON Structure of the response conforms to the [Style JSON Schema](#). The Linked Data formats conforms to the Style Ontology.

### Links

The following table defines the link relation types accessible from a style instance that provide transitions to other states related to the style instance.

Relation	Description
<code>self</code>	Refers to this resource itself
<code>service</code>	Refers to the root of the portrayal service
<code>portrayal:styles</code>	Refers to the style search endpoint
<code>curies</code>	Refers to the curies defined for the links

## Example response

===== HAL+JSON Format Response (Level 3 REST API)

The HAL+JSON response contains the description of the Style in JSON-LD (which can be converted to Linked Data by applying the JSON-LD context of the service). In addition, it provides links to other states that can be followed by clients following the semantic of the link relation types described in the [Style Link section](#).

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 1895

{
  "id": "345a19896b477721787a82742cc04770",
  "uri": "http://www.opengis.net/testbed/12/portrayal/ems/style#EMSIncidentStyle",
  "type": "style:FeatureTypeStyle",
  "title": "EMS Incident Type Style",
  "description": "EMS Incident Type Style",
  "created": "2016-11-18T03:18:12.661Z",
  "modified": "2016-11-18T03:18:15.024Z",
  "featureType": "http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident",
  "rules": [

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.wind.strongWind-
    portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.geophysical.magnet
    icStorm-portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.crime.bombThreat-
    portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.hazardousMaterial.
    biologicalHazard-portrayal-rule",
    .... truncated ....

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.geophysical.earthq
    uake-portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.crime.bombExplosio
    n-portrayal-rule"
  ],
  "_links": {
    "self": {
      "href": "http://localhost:8082/styles/345a19896b477721787a82742cc04770"
    },
    "portrayal:styles": {
      "href": "http://localhost:8082/styles"
    },
  },
  "service": {
```

```

    "href": "http://localhost:8082"
  },
  "curies": [{
    "href": "http://www.opengis.net/rels/portrayal/{rel}",
    "name": "portrayal",
    "templated": true
  }]
}
}

```

=====  
 ===== JSON(-LD) Format Response (Level 2 REST API)

The following HTTP request performs a GET Request to get an instance of a Style in JSON format.

```

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1495

{
  "id": "345a19896b477721787a82742cc04770",
  "uri": "http://www.opengis.net/testbed/12/portrayal/ems/style#EMSIncidentStyle",
  "type": "style:FeatureTypeStyle",
  "title": "EMS Incident Type Style",
  "description": "EMS Incident Type Style",
  "created": "2016-11-18T03:18:12.661Z",
  "modified": "2016-11-18T03:18:15.024Z",
  "featureType": "http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident",
  "rules": [

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.wind.strongWind-
    portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.geophysical.magnet
    icStorm-portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.crime.bombThreat-
    portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.hazardousMaterial.
    biologicalHazard-portrayal-rule",
    .... truncated ....

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.geophysical.earthq
    uake-portrayal-rule",

    "http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.crime.bombExplosio
    n-portrayal-rule"
  ]
}

```

The JSON(LD) response is identical to the HAL+JSON except it does not have the hyperlinks to other states. The client will need to build the URL to reach other states (by reading documentation of API).

===== Turtle Format Response (Linked Data API)

The following HTTP request performs a GET Request to get an instance of a Style in Turtle format.

```
GET /styles/345a19896b477721787a82742cc04770 HTTP/1.1
Accept: text/turtle
Host: localhost
```

The response returns a TTL document that can be processed and interpreted by machine using the Style ontology.

```
HTTP/1.1 200 OK
Content-Type: text/turtle
Content-Length: 10395

@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix style: <http://www.opengis.net/ont/portrayal/style#> .

<http://www.opengis.net/testbed/12/portrayal/ems/style#EMSIncidentStyle>
  a style:FeatureTypeStyle ;
  dcterms:created "2016-11-18T03:18:12.661Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
  dcterms:description "EMS Incident Type Style" ;
  dcterms:modified "2016-11-18T03:18:15.024Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
  dcterms:title "EMS Incident Type Style" ;
  style:featureType
    <http://www.opengis.net/testbed11/ont/incident/ems#EMSIncident> ;
  style:hasRule

  <http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.wind.strongWind-
  portrayal-rule> ,

  <http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.geophysical.magnet
  icStorm-portrayal-rule> ,

  <http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.crime.bombThreat-
  portrayal-rule> ,

  .... truncated.....

  <http://www.opengis.net/testbed/12/portrayal/ems/style#ems.incident.crime.bombExplosio
  n-portrayal-rule> .
```

## Render Geospatial Data (Proposed)

To render a particular dataset, a HTTP **GET** request will be performed on the style renderer endpoint is performed. The request should accept one of the supported graphic formats advertised in the capabilities by the service. The endpoint provides the ability to customize the size of the graphic returned in the response.

### Query Parameters

Parameter	Type	Description	Cardinality
url	String	URL to access geospatial data	
style	String	Identifier of a style (id or URI)	1
bbox	String	Bounding box geospatial data to render in the given CRS	1
crs	String	Coordinate Reference System (default is EPSG:4326)	0..1
width	integer	Width in pixel of the rendered map layer	0..1
height	integer	Height in pixel of the rendered map layer	0..1

### Response Structure

The Response of the render action is according the mime format accepted by the renderer, which is defined in the capabilities resource.

## Symbols

A **Symbol** defines a graphic representation of a feature. The **Symbols resource** is used to search symbol instances based on some search criteria.

### JSON Schema

The following defines the JSON schema for the Symbol derived from the Symbol Ontology defined in Appendix C. The JSON-LD Context of the service can be used to convert the JSON document compliant to this schema to the Linked Data Representation.

Path	Type	Description	Card.
<b>id</b>	<i>String</i>	Internal identifier for the symbol (which can be used in Level 2 API)	1

Path	Type	Description	Card.
uri	String	Linked Data URI for the Symbol (equivalent to @id in JSON-LD)	0..1
type	String	The type (class) of the Symbol (symbol:Symbol or (future) subclasses).	1
title	String	The title of the symbol	1
titleMap	Object	The title map for each language of the title . Each key corresponds to the two letter language identifier name ( for example "en")	0..1
description	String	The description of the item	0..1
descriptionMap	Object	The description map for each language of the description. Each key corresponds to the two letter language identifier name ( for example "en")	0..1
created	String	The date of creation (XSD datetime format) (generated by the service)	1
modified	String	The date of the last modification (XSD datetime format) (generated by the service)	0..1
symbolizers[]	Array	Array of symbolizers used to render the symbol. See ontology for Symbolizer in Appendix C	1

## Search Symbols

The search of the Symbols is performed by performing a HTTP **GET** request on the Symbols resource. The symbol search supports free text, by ids, by uris, CQL constraint and can return results with aggregations on specific facets.

## Query Parameters

The following query parameters are supported in the query:

Parameter	Description	Cardinality
<code>q</code>	Text to search in textual fields	0..1
<code>uri</code>	One or more URIs of Symbol instances	0..n
<code>id</code>	One or more id of Symbol instances	0..n
<code>includeFacet</code>	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.	0..1
<code>facet.fieldname</code>	Constraint values of a given facet field name	0..n
<code>constraint</code>	A constraint expressed in CQL. This is used to express more advanced query filtering	0..1
<code>fields</code>	One or more fields to be included in the response. Use JSON path dot notation for referring paths	0..n
<code>pageNumber</code>	The number of the current page as defined in <a href="#">Paging section</a>	0..1
<code>pageSize</code>	The count of items on the current page as defined in <a href="#">Paging section</a>	0..1
<code>sort</code>	The sorting parameters as defined in <a href="#">Sorting section</a> .	0..n

### Example request

The following performs a symbol search containing the keyword 'Rail'.

```
GET /symbols?q=Rail HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response search is structured according the [Search Results Schema](#). The items of the results conforms to the [Symbol JSON Schema](#). The collection name used in the `_embedded` section of the HAL response is called `portrayal:items`.

### Links

The following link relation types are provided in the response to allow the transition to others states from the symbol search results embedded in the response.

Relation	Description
<code>self</code>	Refers to this resource itself



Relation	Description
<code>service</code>	Refers to the root of the portrayal service
<code>curies</code>	Refers to the curies defined for the links
<code>first</code>	The first page of results
<code>last</code>	The last page of results
<code>next</code>	The next page of results
<code>prev</code>	The previous page of results

### Example response

===== HAL+JSON Response

The embedded objects in the response of the request conforms to the [Symbol JSON Schema](#). The collection name used in the `_embedded` section of the HAL response is called `portrayal:items`. The rest of the response return paging information and links to other states.

```

HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 2707

{
  "_embedded": {
    "portrayal:items": [{
      "id": "41f4b3ed5976016d071cf30498bf6358",
      "uri":
"http://www.opengis.net/testbed/12/hswg/symbols#RailHijackingSymbol",
      "type": "symbol:Symbol",
      "title": "Rail Hijacking",
      "created": "2016-11-17T15:41:35.436Z",
      "modified": "2016-11-17T15:41:35.951Z",
      "_links": {
        "self": {
          "href":
"http://localhost:8082/symbols/41f4b3ed5976016d071cf30498bf6358"
        }
      }
    }, {
      "id": "d6c8d924a629514d322bb25f563f8948",
      "uri":
"http://www.opengis.net/testbed/12/hswg/symbols#RailAccidentSymbol",
      "type": "symbol:Symbol",
      "title": "Rail Accident",
      "created": "2016-11-17T15:42:18.602Z",
      "modified": "2016-11-17T15:42:18.717Z",
      "_links": {
        "self": {
          "href":

```

```

"http://localhost:8082/symbols/d6c8d924a629514d322bb25f563f8948"
    }
  }
}, {
  "id": "d6792c5cb00daa3b9e2c161846cc7ef4",
  "uri":
"http://www.opengis.net/testbed/12/hswg/symbols#RailIncidentSymbol",
  "type": "symbol:Symbol",
  "title": "Rail Incident",
  "created": "2016-11-17T15:42:11.281Z",
  "modified": "2016-11-17T15:42:11.712Z",
  "_links": {
    "self": {
      "href":
"http://localhost:8082/symbols/d6792c5cb00daa3b9e2c161846cc7ef4"
    }
  }
}]
},
"_links": {
  "self": {
    "href": "http://localhost:8082/symbols?q=Rail"
  },
  "service": {
    "href": "http://localhost:8082"
  },
  "curies": [{
    "href": "http://www.opengis.net/rels/portrayal/{rel}",
    "name": "portrayal",
    "templated": true
  }]
},
"aggregations": [],
"page": {
  "size": 20,
  "totalElements": 3,
  "totalPages": 1,
  "number": 0
}
}

```

=====  
 ===== JSON Response

The result objects in the response of the request conforms to the [Symbol JSON Schema](#). The rest of the response return paging information.

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1007
```

```
{
  "results": [{
    "id": "41f4b3ed5976016d071cf30498bf6358",
    "uri": "http://www.opengis.net/testbed/12/hswg/symbols#RailHijackingSymbol",
    "type": "symbol:Symbol",
    "title": "Rail Hijacking",
    "created": "2016-11-17T15:41:35.436Z",
    "modified": "2016-11-17T15:41:35.951Z"
  }, {
    "id": "d6c8d924a629514d322bb25f563f8948",
    "uri": "http://www.opengis.net/testbed/12/hswg/symbols#RailAccidentSymbol",
    "type": "symbol:Symbol",
    "title": "Rail Accident",
    "created": "2016-11-17T15:42:18.602Z",
    "modified": "2016-11-17T15:42:18.717Z"
  }, {
    "id": "d6792c5cb00daa3b9e2c161846cc7ef4",
    "uri": "http://www.opengis.net/testbed/12/hswg/symbols#RailIncidentSymbol",
    "type": "symbol:Symbol",
    "title": "Rail Incident",
    "created": "2016-11-17T15:42:11.281Z",
    "modified": "2016-11-17T15:42:11.712Z"
  }
  ],
  "page": {
    "size": 20,
    "number": 0,
    "totalElements": 3
  }
}
```

## Symbol

The Symbol resource is used to retrieve a Symbol instance.

### Retrieve a Symbol

To retrieve a particular instance of Symbol, a HTTP **GET** request will get the details of a Symbol. There are two ways to retrieve an instance of a schema, using an internal id in the path or by using its Linked Data URL (using uri as query parameter).

### Query Parameters

Parameter	Description	Cardinality
uri	URI of the symbol if the instance needs to be retrieved by URI. URI should be encoded to escape special characters.	0..1

### Example request

The following HTTP request performs a GET Request to get an instance of a symbol identified by its internal id *06aab0337e96ea6e2535a2620b8e9a90*

```
GET /symbols/06aab0337e96ea6e2535a2620b8e9a90 HTTP/1.1
Accept: application/hal+json
Host: localhost
```

The following query get the instance identified by the URI <http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-symbol>

```
GET
/symbols/instance?uri=http%3A%2F%2Fwww.opengis.net%2Ftestbed%2F12%2Fems%2Fsymbols%23ems.incident.wind-symbol HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response of the instance request can be retrieved in HAL+JSON, JSON-LD and Linked Data formats. The JSON Structure of the reponse conforms to the [Symbol JSON Schema](#). The Linked Data formats conform to the Symbol Ontology.

### Links

The following table defines the link relation types accessible from a symbol instance that provide transitions to other states related to the symbol instance.

Relation	Description
<i>self</i>	Refers to this resource itself
<i>service</i>	Refers to the root of the portrayal service
<i>portrayal:symbols</i>	Refers to the symbols search endpoint
<i>portrayal:renderer</i>	Refers to the renderer of the symbol.
<i>curies</i>	Refers to the curies defined for the links

## Example response

===== HAL+JSON Format Response (Level 3 REST API)

The HAL+JSON response contains the description of the Symbol in JSON-LD (which can be converted to Linked Data by applying the JSON-LD context of the service). In addition, it provides links to other states that can be followed by clients following the semantic of the link relation types described in the [Symbol Link section](#).

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 1895

{
  "id": "06aab0337e96ea6e2535a2620b8e9a90",
  "uri": "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-symbol",
  "type": "symbol:Symbol",
  "register": [
    "portrayal"
  ],
  "title": "wind",
  "created": "2016-11-17T15:39:40.261Z",
  "modified": "2016-11-17T15:39:40.711Z",
  "symbolName": "ems.incident.wind",
  "specification": "https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf",
  "symbolSet": "http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet",
  "browseGraphic": {
    "type": "graphic:ExternalGraphic",
    "uri": "http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
    "title": "ems.incident.wind icon",
    "description": "icon for ems.incident.wind",
    "onlineResource":
"http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
    "format": "image/png"
  },
  "symbolizers": [{
    "uri": "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-
pointSymbolizer",
    "type": "symbolizer:PointSymbolizer",
    "title": "PointSymbolizer for wind symbol",
    "graphicSymbol": {
      "type": "graphic:GraphicSymbol",
      "externalGraphic": {
        "type": "graphic:ExternalGraphic",
        "uri":
"http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
        "title": "ems.incident.wind icon",
        "description": "icon for ems.incident.wind",
        "onlineResource":
"http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
        "format": "image/png"
      }
    }
  ]
}
```

```

    }
  },
  "denotes": [
    "http://www.opengis.net/taxonomy/ems#ems.incident.wind"
  ],
  "_links": {
    "self": {
      "href": "http://localhost:8082/symbols/06aab0337e96ea6e2535a2620b8e9a90"
    },
    "portrayal:render": {
      "href":
"http://localhost:8082/symbols/06aab0337e96ea6e2535a2620b8e9a90/render"
    },
    "portrayal:symbols": {
      "href": "http://localhost:8082/symbols"
    },
    "service": {
      "href": "http://localhost:8082"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/portrayal/{rel}",
      "name": "portrayal",
      "templated": true
    }]
  }
}

```

===== JSON(-LD) Format Response (Level 2 REST API)

The following HTTP request performs a GET Request to get an instance of a Symbol in JSON format.

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: 1495

```
{
  "id": "06aab0337e96ea6e2535a2620b8e9a90",
  "uri": "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-symbol",
  "type": "symbol:Symbol",
  "register": [
    "portrayal"
  ],
  "title": "wind",
  "created": "2016-11-17T15:39:40.261Z",
  "modified": "2016-11-17T15:39:40.711Z",
  "symbolName": "ems.incident.wind",
  "specification": "https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf",
  "symbolSet": "http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet",
  "browseGraphic": {
    "type": "graphic:ExternalGraphic",
    "uri": "http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
    "title": "ems.incident.wind icon",
    "description": "icon for ems.incident.wind",
    "onlineResource":
"http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
    "format": "image/png"
  },
  "symbolizers": [{
    "uri": "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-
pointSymbolizer",
    "type": "symbolizer:PointSymbolizer",
    "title": "PointSymbolizer for wind symbol",
    "graphicSymbol": {
      "type": "graphic:GraphicSymbol",
      "externalGraphic": {
        "type": "graphic:ExternalGraphic",
        "uri":
"http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
        "title": "ems.incident.wind icon",
        "description": "icon for ems.incident.wind",
        "onlineResource":
"http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png",
        "format": "image/png"
      }
    }
  }],
  "denotes": [
    "http://www.opengis.net/taxonomy/ems#ems.incident.wind"
  ]
}
```

The JSON-(LD) response is identical to the HAL+JSON except it does not have the hyperlinks to other states. The client will need to build the URL to reach other states (by reading documentation of API).

===== Turtle Format Response (Linked Data API)

The following HTTP request performs a GET Request to get an instance of a Symbol in Turtle format.

```
GET /symbols/06aab0337e96ea6e2535a2620b8e9a90 HTTP/1.1
Accept: text/turtle
Host: localhost
```

The response returns a TTL document that can processed and interpreted by machine using the Portrayal ontology.



HTTP/1.1 200 OK

Content-Type: text/turtle

Content-Length: 1395

@prefix symbol: <http://www.opengis.net/ont/portrayal/symbol#> .

@prefix dcterms: <http://purl.org/dc/terms/> .

<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-symbol>

a symbol:Symbol ;

dcterms:title "wind" ;

symbol:browseGraphic

<http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png> ;

symbol:specification <https://cms.masas-

x.ca.s3.amazonaws.com/EMS\_Symbology\_v1.0.pdf> ;

symbol:symbolName "ems.incident.wind" ;

symbol:symbolSet

<http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet> ;

<http://www.opengis.net/ont/portrayal/symbolizer#symbolizer>

<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-pointSymbolizer> .

<http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png>

a

<http://www.opengis.net/ont/portrayal/graphic#ExternalGraphic> ;

dcterms:description "icon for ems.incident.wind" ;

dcterms:title "ems.incident.wind icon" ;

<http://www.opengis.net/ont/portrayal/graphic#format>

"image/png" ;

<http://www.opengis.net/ont/portrayal/graphic#onlineResource>

"http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png"

.

<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind-pointSymbolizer>

a

<http://www.opengis.net/ont/portrayal/symbolizer#PointSymbolizer> ;

dcterms:title "PointSymbolizer for wind symbol" ;

<http://www.opengis.net/ont/portrayal/graphic#graphicSymbol>

[ a <http://www.opengis.net/ont/portrayal/graphic#GraphicSymbol>

;

<http://www.opengis.net/ont/portrayal/graphic#externalGraphic>

<http://ows.usersmarts.com/ems/icons/tier1/Base/ems.incident.wind.png>

] .

## Symbol Renderer

The Symbol Renderer resource is used to render a symbol into a graphic representation such as SVG, PNG, TIFF, JPEG using its symbolizer definitions. This endpoint provides the ability to render the symbol into different formats and size.

## Render a Symbol

To render a particular instance of Symbol, a HTTP **GET** request will get on the symbol renderer endpoint is performed. The request should accept one of the supported graphic formats advertised in the capabilities by the service. The endpoint provides the ability to customize the size of the graphic returned in the response.

### Query Parameters

Parameter	Type	Description	Cardinality
width	integer	Width in pixel of the rendered symbol	0..1
height	integer	Height in pixel of the rendered symbol	0..1

### Response Structure

The Response of the render action is according the mime format accepted by the renderer, which is defined in the capabilities resource.

### Examples

The following example requests the renderer to render the symbol Wind Incident to a PNG image of size 512x 512.

```
GET /symbols/06aab0337e96ea6e2535a2620b8e9a90/render?width=512&height=512 HTTP/1.1
Accept: image/png
Host: localhost
```

The image produces is the following:



## SymbolSets

A **SymbolSet** defines a set of Symbols. The **SymbolSets resource** is used to search SymbolSet instances based on some search criteria.

### JSON Schema

The following defines the JSON schema for the SymbolSet derived from the Symbol Ontology defined in Appendix C. It can also be derived from the JSON-LD Context

Path	Type	Description	Card.
<code>id</code>	<code>String</code>	Internal identifier for the SymbolSet (which can be used in Level 2 API)	1
<code>uri</code>	<code>String</code>	Linked Data URI for the SymbolSet (equivalent to <code>@id</code> in JSON-LD)	0..1
<code>type</code>	<code>String</code>	The type (class) of the SymbolSet ( <code>symbol:SymbolSet</code> ).	1

Path	Type	Description	Card.
title	String	The title of the symbol	1
titleMap	Object	The title map for each language of the title . Each key corresponds to the two letter language identifier name ( for example "en")	0..1
description	String	The description of the item	0..1
descriptionMap	Object	The description map for each language of the description. Each key corresponds to the two letter language identifier name ( for example "en")	0..1
created	String	The date of creation (XSD datetime format) (generated by the service)	1
modified	String	The date of the last modification (XSD datetime format) (generated by the service)	0..1
specification	URL	The reference to a specification document	0..1
symbols[]	Array	Array of symbols URIs.	0..1

## Search SymbolSets

The search of the SymbolSets is performed by performing a HTTP **GET** request on the SymbolSets resource. The symbolSet search supports free text, by ids, by uris, CQL constraint and can return results with aggregations on specific facets.

### Query Parameters

The following query parameters are supported in the query:

Parameter	Description	Cardinality
q	Text to search in textual fields	0..1
uri	One or more URIs of Symbol instances	0..n

Parameter	Description	Cardinality
<code>id</code>	One or more id of Symbol instances	0..n
<code>includeFacet</code>	Boolean or list of facet names to include for aggregation computation. If the value is true, include all facets supported by the server. If only a subset of the facets are needed, a comma delimited of field names can be set.	0..1
<code>facet.fieldname</code>	Constraint values of a given facet field name	0..n
<code>constraint</code>	A constraint expressed in CQL. This is used to express more advanced query filtering	0..1
<code>fields</code>	One or more fields to be included in the response. Use JSON path dot notation for referring paths	0..n
<code>pageNumber</code>	The number of the current page as defined in <a href="#">Paging section</a>	0..1
<code>pageSize</code>	The count of items on the current page as defined in <a href="#">Paging section</a>	0..1
<code>sort</code>	The sorting parameters as defined in <a href="#">Sorting section</a> .	0..n

### Example request

The following performs a `symbolSet` search containing the keyword 'Rail'.

```
GET /symbolsets HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response search is structured according the [Search Results Schema](#). The items of the results conform to the [SymbolSet JSON Schema](#). The collection name used in the `_embedded` section of the HAL response is called `portrayal:items`.

### Links

The following link relation types are provided in the response to allow the transition to others states from the `symbolSet` search results embedded in the response.

Relation	Description
<code>self</code>	Refers to this resource itself
<code>service</code>	Refers to the root of the portrayal service
<code>curies</code>	Refers to the curies defined for the links
<code>first</code>	The first page of results
<code>last</code>	The last page of results
<code>next</code>	The next page of results

Relation	Description
<a href="#">prev</a>	The previous page of results

### Example response

===== HAL+JSON Response

The embedded objects in the response of the request conforms to the [SymbolSet JSON Schema](#). The collection name used in the `_embedded` section of the HAL response is called `portrayal:items`. The rest of the response return paging information and links to other states.

```

HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 2707

{
  "_embedded": {
    "portrayal:items": [{
      "id": "4cabf46007df9b652151927033e265c6",
      "uri": "http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet",
      "type": "symbol:SymbolSet",
      "description": "Standard Canadian Emergency Mapping Symbology (EMS)
SymbolSet version 1.0",
      "created": "2016-11-17T15:38:35.856Z",
      "modified": "2016-11-17T15:38:36.457Z",
      "_links": {
        "self": {
          "href":
"http://localhost:8082/symbolsets/4cabf46007df9b652151927033e265c6"
        }
      }
    }, {
      "id": "bf83c8b6f4033ffd4c93a2be14ebe476",
      "uri": "http://www.opengis.net/testbed/12/hswg/symbols#HSWGSymbolSet",
      "type": "symbol:SymbolSet",
      "description": "Home Security Working Group (HSWG) SymbolSet version 1.0",
      "created": "2016-11-17T15:38:37.357Z",
      "modified": "2016-11-17T15:38:37.786Z",
      "_links": {
        "self": {
          "href":
"http://localhost:8082/symbolsets/bf83c8b6f4033ffd4c93a2be14ebe476"
        }
      }
    }
  ]
},
  "_links": {
    "self": {
      "href": "http://localhost:8082/symbolsets"
    }
  }
}

```

```
    },
    "service": {
      "href": "http://localhost:8082"
    },
    "curies": [{
      "href": "http://www.opengis.net/rels/portrayal/{rel}",
      "name": "portrayal",
      "templated": true
    }]
  },
  "aggregations": [],
  "page": {
    "size": 20,
    "totalElements": 2,
    "totalPages": 1,
    "number": 0
  }
}
```

===== JSON Response

The result objects in the response of the request conforms to the [SymbolSet JSON Schema](#). The rest of the response return paging information.

```

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 850

{
  "results": [{
    "id": "4cabf46007df9b652151927033e265c6",
    "uri": "http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet",
    "type": "symbol:SymbolSet",
    "description": "Standard Canadian Emergency Mapping Symbology (EMS) SymbolSet
version 1.0",
    "created": "2016-11-17T23:13:47.142Z",
    "modified": "2016-11-17T23:13:48.534Z"
  }, {
    "id": "bf83c8b6f4033ffd4c93a2be14ebe476",
    "uri": "http://www.opengis.net/testbed/12/hswg/symbols#HSWGSymbolSet",
    "type": "symbol:SymbolSet",
    "description": "Home Security Working Group (HSWG) SymbolSet version 1.0",
    "created": "2016-11-17T23:13:50.354Z",
    "modified": "2016-11-17T23:13:50.922Z"
  }],
  "page": {
    "size": 20,
    "number": 0,
    "totalElements": 2
  }
}

```

## SymbolSet

The SymbolSet resource is used to retrieve a SymbolSet instance.

### Retrieve a SymbolSet

To retrieve a particular instance of SymbolSet, a HTTP **GET** request will get the details of a SymbolSet. There are two ways to retrieve an instance of a SymbolSet, using an internal id in the path or by using its Linked Data URL (using uri as query parameter).

#### Query Parameters

Parameter	Description	Cardinality
uri	URI of the symbol if the instance needs to be retrieved by URI. URI should be encoded to escape special characters.	0..1

#### Example request

The following HTTP request performs a GET Request to get an instance of a symbol identified by its



internal id *4cabf46007df9b652151927033e265c6*

```
GET /symbolsets/4cabf46007df9b652151927033e265c6 HTTP/1.1
Accept: application/hal+json
Host: localhost
```

The following query get the instance identified by the URI <http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet>

```
GET
/symbolsets/instance?uri=http%3A%2F%2Fwww.opengis.net%2Ftestbed%2F12%2Fems%2Fsymbols%2
3EMSSymbolSet HTTP/1.1
Accept: application/hal+json
Host: localhost
```

### Response structure

The response of the instance request can be retrieved in HAL+JSON, JSON-LD and Linked Data formats. The JSON Structure of the response conforms to the [SymbolSet JSON Schema](#). The Linked Data formats conforms to the Symbol Ontology.

### Links

The following table defines the link relation types accessible from a symbolSet instance that provide transitions to other states related to the symbolSet instance.

Relation	Description
<a href="#">self</a>	Refers to this resource itself
<a href="#">service</a>	Refers to the root of the portrayal service
<a href="#">portrayal:symbolSets</a>	Refers to the symbols search endpoint
<a href="#">curies</a>	Refers to the curies defined for the links

### Example response

===== HAL+JSON Format Response (Level 3 REST API)

The HAL+JSON response contains the description of the SymbolSet in JSON-LD (which can be converted to Linked Data by applying the JSON-LD context of the service). In addition, it provides links to other states that can be followed by clients following the semantic of the link relation types described in the [SymbolSet Link section](#).

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 1895

{
```

```

    "id": "4cabf46007df9b652151927033e265c6",
    "uri": "http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet",
    "type": "symbol:SymbolSet",
    "register": [
      "portrayal"
    ],
    "description": "Standard Canadian Emergency Mapping Symbology (EMS) SymbolSet
version 1.0",
    "created": "2016-11-17T15:38:35.856Z",
    "modified": "2016-11-17T15:38:36.457Z",
    "specification": "https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf",
    "symbols": [
      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.temperature-
symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.railway.railwayAccident-
symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.crime.industrialCrime-
symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.hazardousMaterial.infectio
usDisease-symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind.hurricaneForceWind-
symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.missingPerson.silver-
symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.hazardousMaterial.poisonou
sGas-symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.meteorological.hurricane-
symbol",
      .... (truncated) ....
      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.crime.retailCrime-
symbol",

      "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.animalHealth.animalDieOff-
symbol"
    ],
    "_links": {
      "self": {
        "href":
"http://localhost:8082/symbolsets/4cabf46007df9b652151927033e265c6"
      },
      "portrayal:symbolSets": {
        "href": "http://localhost:8082/symbolsets"
      },
    },
    "service": {

```

```
    "href": "http://localhost:8082"
  },
  "curies": [{
    "href": "http://www.opengis.net/rels/portrayal/{rel}",
    "name": "portrayal",
    "templated": true
  }]
}
}
```

==== JSON(-LD) Format Response (Level 2 REST API)

The following HTTP request performs a GET Request to get an instance of a Symbol in JSON format.

HTTP/1.1 200 OK

Content-Type: application/json

Content-Length: 1495

```
{
  "id": "4cabf46007df9b652151927033e265c6",
  "uri": "http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet",
  "type": "symbol:SymbolSet",
  "register": [
    "portrayal"
  ],
  "description": "Standard Canadian Emergency Mapping Symbology (EMS) SymbolSet
version 1.0",
  "created": "2016-11-17T15:38:35.856Z",
  "modified": "2016-11-17T15:38:36.457Z",
  "specification": "https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf",
  "symbols": [
    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.temperature-
symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.railway.railwayAccident-
symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.crime.industrialCrime-
symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.hazardousMaterial.infectio
usDisease-symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.wind.hurricaneForceWind-
symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.missingPerson.silver-
symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.hazardousMaterial.poisonou
sGas-symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.meteorological.hurricane-
symbol",
    .... (truncated) ....
    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.crime.retailCrime-
symbol",

    "http://www.opengis.net/testbed/12/ems/symbols#ems.incident.animalHealth.animalDieOff-
symbol"
  ]
}
```

The JSON-LD response is identical to the HAL+JSON except it does not have the hyperlinks to other

states. The client will need to build the URL to reach other states (by reading documentation of API).

===== Turtle Format Response (Linked Data API)

The following HTTP request performs a GET Request to get an instance of a SymbolSet in Turtle format.

```
GET /symbolsets/4cabf46007df9b652151927033e265c6 HTTP/1.1
Accept: text/turtle
Host: localhost
```

The response returns a TTL document that can be processed and interpreted by machine using the Portrayal ontology.

```
HTTP/1.1 200 OK
Content-Type: text/turtle
Content-Length: 1395

<http://www.opengis.net/testbed/12/ems/symbols#EMSSymbolSet>
  a      <http://www.opengis.net/ont/portrayal/symbol#SymbolSet> ;
  <http://purl.org/dc/terms/description>
    "Standard Canadian Emergency Mapping Symbology (EMS) SymbolSet version
1.0" ;
  <http://www.opengis.net/ont/portrayal/symbol#specification>
    "https://cms.masas-x.ca.s3.amazonaws.com/EMS_Symbology_v1.0.pdf" ;
  <http://www.opengis.net/ont/portrayal/symbol#symbol>

<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.temperature-symbol> ,
<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.railway.railwayAccident-
symbol> ,
<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.crime.industrialCrime-
symbol>,
    .... truncated ...

<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.crime.retailCrime-symbol>
,
<http://www.opengis.net/testbed/12/ems/symbols#ems.incident.animalHealth.animalDieOff-
symbol> .
```

## SPARQL Service

The Semantic Portrayal Service provides a SPARQL service endpoint, which implements the [SPARQL Protocol](#), that can accept a SPARQL query on portrayal managed by the service. Both HTTP GET and HTTP POST are supported.

### Query Parameters



- A RDF graph serialized, in the RDF/XML syntax or an equivalent RDF graph serialization (for SPARQL Query forms DESCRIBE and CONSTRUCT). The content type of the response to a successful query operation must be the media type defined for the format of the response body.

### Example response

The following example shows a SPARQL query response

```
HTTP/1.1 200 OK
Content-Type: application/sparql-results+json
Content-Length: 2352
{
  "head": {
    "vars": [ "symbol" , "symbolName" ]
  } ,
  "results": {
    "bindings": [
      {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.airQuality-symbol" } ,
        "symbolName": { "type": "literal" , "value": "ems.incident.airQuality" }
      } ,
      {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.animalHealth-symbol" } ,
        "symbolName": { "type": "literal" , "value": "ems.incident.animalHealth" }
      } ,
      {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.animalHealth.animalDieOff-
symbol" } ,
        "symbolName": { "type": "literal" , "value":
"ems.incident.animalHealth.animalDieOff" }
      } ,
      {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.animalHealth.animalFeed-
symbol" } ,
        "symbolName": { "type": "literal" , "value":
"ems.incident.animalHealth.animalFeed" }
      } ,
      {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.aviation-symbol" } ,
        "symbolName": { "type": "literal" , "value": "ems.incident.aviation" }
      } ,
      {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.aviation.aircraftCrash-
symbol" } ,
```

```

    "symbolName": { "type": "literal" , "value":
"ems.incident.aviation.aircraftCrash" }
    } ,
    {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.aviation.aircraftHijacking
-symbol" } ,
        "symbolName": { "type": "literal" , "value":
"ems.incident.aviation.aircraftHijacking" }
    } ,
    {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.aviation.airportClosure-
symbol" } ,
        "symbolName": { "type": "literal" , "value":
"ems.incident.aviation.airportClosure" }
    } ,
    {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.aviation.airspaceClosure-
symbol" } ,
        "symbolName": { "type": "literal" , "value":
"ems.incident.aviation.airspaceClosure" }
    } ,
    {
        "symbol": { "type": "uri" , "value":
"http://www.opengis.net/testbed/12/ems/symbols#ems.incident.aviation.noticeToAirmen-
symbol" } ,
        "symbolName": { "type": "literal" , "value":
"ems.incident.aviation.noticeToAirmen" }
    }
]
}
}

```



# Appendix G: Revision History

Table 102. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
March 10, 2016	S. Fella	0.1	abstract	initial version
April 10, 2016	S. Fella	0.2	all	initial version
April 14, 2016	S. Fella	0.3	Registry Model	Registry Model
May 31, 2016	S. Fella	0.4		Future work section
June 27, 2016	S. Fella	0.4		All sections
June 27, 2016	S. Fella	0.5		All sections
June 27, 2016	S. Fella	0.56		All sections
October 4, 2016	S. Fella	0.7		Registry REST API
October 10, 2016	S. Fella	0.8		All Sections
October 25, 2016	S. Fella	0.9		All Sections
October 30, 2016	S. Fella	0.10		All Sections
November 3, 2016	S. Fella	0.11		Appendix C
November 7, 2016	S. Fella	0.12		Mediation REST API
November 14, 2016	S. Fella	0.13		Portrayal REST API
November 20, 2016	S. Fella	0.14		Registry REST API
November 20, 2016	S. Fella	0.15		All Sections (formatting)
March 14, 2017	L. Bermudez	0.16		Preface, Future Work and intro

# Appendix H: Bibliography

- [1] OGC, OGC Testbed 11 Demonstration. (2015).
- [2] Testbed-11 Symbology Mediation Engineering Report, OGC document 15-058
- [3] Implementing Linked Data and Semantically Enabling OGC Services Engineering Report, OGC document 15-054
- [4] CCI Ontology Engineering Report, OGC document 14-049
- [5] Guidelines for Successful OGC Interface Standards, OGC document 00-014r1
- [6] SKOS Reference, W3C Recommendation, 18 August 2009. Latest version available at <http://www.w3.org/TR/skos-reference>.
- [7] 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/>
- [8] 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.
- [9] 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.
- [10] Stuckenschmidt, Heiner, Christine Parent, and Stefano Spaccapietra, eds. *Modular ontologies: concepts, theories and techniques for knowledge modularization*. Vol. 5445. Springer, 2009.