DGIWG – 118

Portrayal Registry Service Interface Specification

Document Identifier:	STD-DP-13-001-ed1.0.0-Portrayal_Registry_Service
Publication Date:	11 April 2013
Edition:	1.0.0
Edition Date:	11 April 2013
Responsible Party:	DGIWG
Audience:	DGIWG participants and associates
Abstract:	The main part of this specification provides an interface-independent information model for a Portrayal Registry. Annex A specifies how the information model can be mapped to the CSW ebRIM registry interface. Annex B specifies a RESTful interface to the model.
Copyright:	(C) Copyright DGIWG, some rights reserved - (CC) (By:) Attribution You are free: - to copy, distribute, display, and perform/execute the work - to make derivative works - to make commercial use of the work Under the following conditions: - (By:) Attribution. You must give the original author (DGIWG) credit. - For any reuse or distribution, you must make clear to others the license terms of this work. Any of these conditions can be waived if you get permission from the copyright holder DGIWG. Your fair use and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full license is available from Creative Commons <http://creativecommons.org/licenses/by/2.0/ >).

Table of Contents

List of Figures

List of Tables

i. Document points of contact

All questions regarding this document shall be directed to the editor (secretariat@dgiwg.org) or the contributor organisations:

ii. Contributing nations and organizations

Nation	Organization
France	IGN on behalf of the French Ministry of Defense; Délégation Générale pour l'Armement
Germany	Bundeswehr Geoinformation Office
Sweden	FMV, Swedish Defence Materiel Administration; Carmenta AB, on behalf of FMV
United States	National Geospatial-Intelligence Agency

iii. Revision history

Date	Release	Editors	Primary clauses modified	Description

iv. Future work

The main purpose of this document is to define the discovery and retrieval interface to the portrayal registry that is being developed by DGIWG. The utility of a portrayal registry would benefit a much wider audience than just DGIWG. DGIWG intends to reach out to other communities of interest that need to manage symbology. Requirements from these other communities may require expansion of the capabilities provided by the registry service.

DGIWG is also developing data product specifications for products required by the NGIF project. These specifications include product finishing rules, generalization rules, labelling rules, annotations etc. required to generate a product. These additional rule types will eventually be hosted in the DGIWG portrayal registry and will need to be accessible and discoverable from a portrayal registry service. This will require extensions to the data model defined in this specification.

Tools are needed for efficient management of registry content. The Register Management Tool developed by DGIWG may be one possible solution. Such management aspects are in scope of ongoing DGIWG projects.

This specification will be amended to make the URNs used in Annex A conform to the structure defined in the document “DGIWG Technical Specification for the Usage of the DGIWG Namespace Identifier (NID)” once this document is adopted by DGIWG.

Introduction

This specification has been developed by the DGIWG S01 project. This work has taken place in the context of the development of the concept of a service-oriented architecture (SOA) for geospatial portrayal, based on web-based interfaces built on top of portrayal registries.

A portrayal registry is defined as a managed collection of pieces of information, describing visualization data that can be used in the portrayal of geographic data. The purpose of a registry is to gather the portrayal data, and metadata describing it, in a central location, to allow distributed systems of applications to apply a common visualization to their maps and other geographic data. The registry allows a central authority to manage updates and maintenance to the portrayal data, and metadata, without having to send patches to the various applications consuming the portrayal data.

To benefit from a geospatial portrayal registry in a SOA, customers must use web based service interfaces to discover and retrieve (for subsequent rendering) symbols and portrayal rules held in the portrayal registry.

This specification defines a platform-independent information model that consists of a querying model and operations to discover and retrieve portrayal rules and symbols held in the registry. It accommodates various use cases:

- Discovery and retrieval of portrayal rule sets and symbol sets for specific application schemas;
- Discovery and retrieval of rules and/or symbols for e.g. specific feature types, or delineation.
- Discovery and retrieval of standardized fonts and colours.

The specification further defines two service interfaces implementing the information model that rely on general IT practices and Open Geospatial Consortium Standards. The first one is defined as an extension package of the OGC CSW-ebRIM standard; the second is based on RESTful principles. Communities can choose to implement the interface most suited to their practices and software environment.

From an application perspective, the portrayal registry service will return the requested symbols to the end-user, where his/her application can render the symbols on a display screen, or output a map to a printer. Alternatively, the DGIWG Portrayal Registry may be a symbol and style provider chained with a feature portrayal service that returns a map image to the end-user client. It is also possible to store symbol libraries locally if the application is working in a non-networked environment. The registry service and portrayal service may in fact reside on the local system.

1. Scope

The main part of this specification provides an interface-independent information model for the discovery of a Portrayal Registry. It is structured in several parts:

- Clause 2 specifies the conformance requirements;
- Clause 3 identifies the normative references;
- Clause 4 defines the terms and abbreviations used throughout the document;
- Clause 5 explains the base concepts of the portrayal registry service;
- Clause 6 specifies the portrayal registry information model;
- Clause 7 specifies the required retrieval operations for a portrayal registry;
- Clause 8 defines the abstract test suite for the information model;
- Clause 9 lists use cases covered by the information model.

This specification focuses on discovery and retrieval of the content of the portrayal registry. It does not cover registration aspects of portrayal items.

Additionally, two annexes define implementations of the information model. Annex A specifies how the information model can be mapped to the CSW ebRIM registry interface. Annex B specifies a RESTful interface to the model.

2. Conformance

This specification contains two interfaces that can be implemented. These two interfaces are described in CSW-ebRIM Interface Specification and RESTful Interface Specification. A conformant implementation shall pass the Abstract Test Suite of at least one of these interfaces.

All conformant implementations shall also pass the main Abstract Test Suite specified in chapter 8.

3. Normative References

[CSW-ebRIM-part1] *CSW-ebRIM Registry Service - Part 1: ebRIM profile of CSW* (1.0.1), R. Martell, available at *http://portal.opengeospatial.org/files/?artifact_id=31137*

[CSW-ebRIM-part2] *CSW-ebRIM Registry Service - Part 2: Basic extension package* (1.0.1), R. Martell, available at *http://portal.opengeospatial.org/files/?artifact_id=31138*

[CSW-ebRIM-part3] *CSW-ebRIM Registry Service - Part_3: Abstract Test Suite* (1.0.1), R. Martell, available at *https://portal.opengeospatial.org/files/?artifact_id=31139*

[ebRIM] *ebXML Registry Information Model Version 3.0*, Sally Fuger, Farrukh Najmi, Nikola Stojanovic, available at *http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rim-3.0-os.pdf*

[HTML] HTML5, *A vocabulary and associated APIs for HTML and XHTML*, W3C Candidate Recommendation 17 December 2012, Robin Berjon, Travis Leithead, Erika Doyle Navara, Edward O'Connor, Silvia Pfeiffer, available at http://www.w3.org/TR/html5/

[ISO 19101] ISO 19101:2002, *Geographic information -- Reference model*, available at *http://www.iso.org/*

[ISO 19117] ISO 19117:2012, *Geographic information -- Portrayal*, available at *http://www.iso.org/*

[RFC2045] IETF RFC 2045 (November 1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, Freed, N. and Borenstein, N., available at *http://www.ietf.org/rfc/rfc2045.txt*

[RFC2396] IETF RFC 2396 (August 1998), *Uniform Resource Identifiers (URI): Generic Syntax*, Berners-Lee, T., Fielding, N., and Masinter, L., available at *http://www.ietf.org/rfc/rfc2396.txt*

[RFC2616] IETF RFC 2616 (June 1999), *Hypertext Transfer Protocol – HTTP/1.1*, Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T., available at *http://www.ietf.org/rfc/rfc2616.txt*

[RFC4627] IETF RFC 4627 (July 2006), *The application/json Media Type for JavaScript Object Notation (JSON)*, D. Crockford, available at *http://tools.ietf.org/html/rfc4627*

[RFC6288] IETF RFC 6288 (August 2011), *URN Namespace for the Defence Geospatial Information Working Group (DGIWG)*, C. Reed, available at *http://tools.ietf.org/html/rfc6288*

[RFC6570] IETF RFC 6570 (March 2012), *URI Template*, Google, R. Fielding, Adobe, M. Hadley, MITRE, M. Nottingham, Rackspace, D. Orchard, Salesforce.com, available at *http://tools.ietf.org/html/rfc6570*

[UML] UML Specification 2.0, *Unified Modelling Language (UML) 2.0,* Object Management Group, available at *http://www.uml.org/#UML2.0*

[XLink] *XML Linking Language (XLink) Version 1.0*, World Wide Web Consortium Recommendation, Steve DeRose, Brown University Scholarly Technology Group, Eve

Maler, Sun Microsystems, David Orchard, Jamcracker, available at *http://www.w3.org/TR/xlink/*

[XML] XML 1.0, *Extensible Markup Language (XML) 1.0*, World Wide Web Consortium Recommendation, Bray, T., Paoli, J., Sperberg-McQueen, C.M., and Maler, E., eds., available at *http://www.w3.org/TR/xml/*

[XSD] XML Schema, *XML Schema Part 1: Structures*, World Wide Web Consortium Recommendation, Thompson, H.S., Beech, D., Maloney, M., and Mendelsohn, N., eds., available at *http://www.w3.org/TR/xmlschema-1/*

4. Terms, definitions and abbreviations

4.1 Terms and definitions

4.1.1 Application schema

Conceptual schema for data required by one or more applications.

[ISO 19101]

NOTE: The portrayal registry information model only contains the information needed to associate a rule set to a data set, as well as associating rules in the rule set to specific feature types. Metadata enables the retrieval of the full application schema definition from the authoritative source.

4.1.2 CSW-ebRIM Extension Package

A principle for adapting the CSW-ebRIM profile by means of a registry package containing a cohesive set of registry objects and related artifacts that extend the ebRIM core information model to meet the needs of some application domain or community of practice.

4.1.3 Data

A discoverable and retrievable digital resource.

4.1.4 ebRIM

XML schema for defining the information model of a registry.

4.1.5 Extrinsic Object

ebRIM representation of a registry object that describes a repository item managed by the registry.

4.1.6 Feature Portrayal Service

Map service capable of retrieving and applying portrayal information (rules and associated symbols) that are referenced in requests sent to the service.

EXAMPLE: An FPS can be a WMS that retrieves visualization instructions referenced via OnlineResources in an SLD document.

4.1.7 Feature type

Categorization of features that share common characteristics expressed as attributes and operations.

EXAMPLE: “Road” feature type carrying a “name” attribute and a “length()” operation.

NOTE: Rules are symbolically mapped to relevant layers by the client, who must declare what feature types the layers of a map contain.

4.1.8 Metadata

An indexed piece of information describing some **data**.

4.1.9 Portrayal Registry

Collection of indexed pieces of portrayal information, which acts as a central authority and can be accessed by clients to retrieve visualization data, enabling the clients to produce a consistent visualization across distributed system domains.

4.1.10 Registry

Index of the **repository**, with some additional information about the **repository item**s, i.e. the **metadata**.

NOTE 1: Depending on the interface used when implementing a portrayal registry, this distinction may, or may not, be relevant to the end user.

NOTE 2: This corresponds to the OASIS ebXML Registry Service specification (bibliography [3]) definition of a registry, not the ISO 19135 (bibliography [8]) definition.

4.1.11 Repository

Collection of **data** – in this case – pieces of portrayal information.

4.1.12 Repository item

Item in the **repository**. The item may be a file, a row or a collection of rows in a database or an object in any other kind of storage medium.

4.1.13 REST

Software architecture used to create stateless web services.

4.1.14 RESTful

Conforming to the **REST** software architecture principles.

4.1.15 Rule

A function that maps geospatial features to symbols.

NOTE 1: This definition corresponds to the [ISO 19117] definition of a portrayal function.

NOTE 2: Rules typically contain a set of conditions that determine the symbols to use when visualizing specific features. Since rules may use attribute lookup to determine the correct symbol for individual features, they can be specific to a certain application schema or feature catalog.

NOTE 3: Rule data may be encoded as SE (bibliography [6]) or any other suitable format.

4.1.16 Rule set

Collection of rules that together make up a meaningful portrayal, usually restricted to a certain application schema.

NOTE: This corresponds to the [ISO 19117] concept of a portrayal function set: “a function that maps a feature catalogue to a symbol set”.

4.1.17 Symbol

Piece of information that can be used to define the visualization of geometries.

NOTE 1: Symbol data may be encoded as SE (bibliography [6]) symbolizers or any other suitable format.

NOTE 2: This is a specialization of the [ISO 19117] symbol definition – “portrayal primitive that can be graphic, audible, or tactile in nature, or a combination of these”.

4.1.18 Symbol set

Collection of symbols.

[ISO 19117]

NOTE: It is typically used to group a set of related symbols.
EXAMPLE: All symbols in the S-52 standard for naval chart presentation.

4.2 Abbreviations

CSW	Catalog Service for the Web
CSW-ebRIM	ebRIM Profile of CSW – cf. also [CSW-ebRIM-part1]
ebRIM	eBusiness XML Registry Information Model – cf. also [ebRIM]
FPS	Feature Portrayal Service
HTTP	Hypertext Transfer Protocol
OCL	Object Constraint Language
OGC	Open Geospatial Consortium
REST	Representational State Transfer
SE	Symbology Encoding – cf. also bibliography [6].
SLD	Styled Layer Descriptor – cf. also bibliography [4] and [5].
URN	Uniform Resource Name
XML	Extensible Markup Language – cf. also [XML]

5. Central concepts

This chapter contains concise explanations of concepts that are of importance to this specification.

5.1 Data and metadata

Throughout this document, the distinction is made between objects' data and their metadata, e.g. *rule data* and *rule metadata*. *Rule data* refers to the repository content; the actual rules, in what encoding they may be expressed, stored in the repository. *Rule metadata* refers to the registry entries concerning the rule and contains only metadata about rules, such as who created it, what it is intended for etc. Note that the rule metadata does not contain any conditions.

This specification only specifies the information model of the *metadata*, and restricts the *data* in no other way than that it must be possible to express its encoding as a MIME type (cf. [RFC2045]). This MIME type is accessible through the *nativeEncoding* attribute on various objects in the information model.

5.1.1 MIME type recommendations

Some of the common portrayal-related formats have no officially registered MIME types. To facilitate interoperability in spite of this, the table below contains recommendations on the MIME types to use for these formats.

Table 1 – MIME type recommendations

Format	MIME type
OGC Styled Layer Descriptor document, version 1.0	application/vnd.ogc.sld+xml; version 1.0
OGC Styled Layer Descriptor document, version 1.1	application/vnd.ogc.sld+xml; version 1.1
OGC Symbology Encoding document, version 1.1	application/vnd.ogc.se+xml; version 1.1

If officially registered MIME types become available for any of the formats in the table above, these official MIME types take precedence over the recommendations in this specification.

6. Portrayal Registry information model

This chapter describes the classes needed to implement a portrayal registry that exposes a meaningful structure to an end user. Since this is not an interface specification, the data types of the class attributes are not defined. It is up to each separate interface specification to specify types and other interface specific requirements.

The information model is designed to be independent of the underlying format used to store the actual portrayal information.

Class attributes are optional unless otherwise specified.

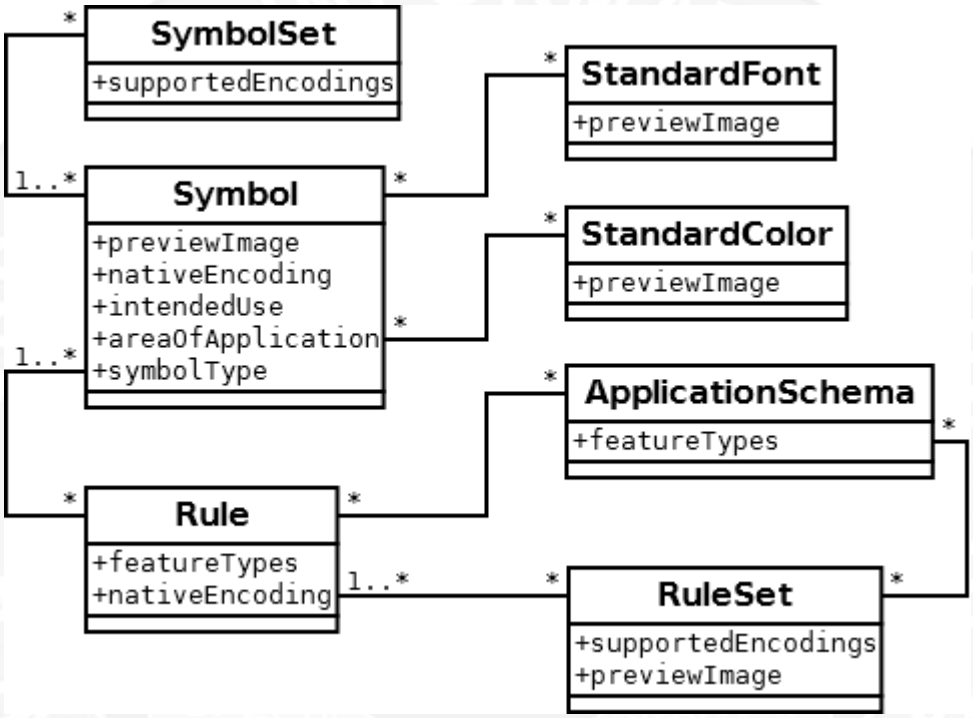


Figure 1: UML diagram of the classes and their relations

6.1 Attributes that are common to all classes

The Base object contains the properties that are common to all classes in the information model.

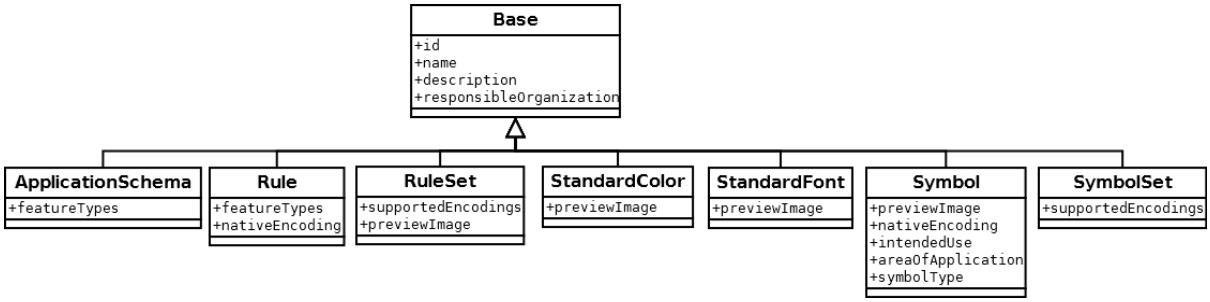


Figure 2: The Base object, its attributes and inheritance

6.1.1 Attributes

- **id** (mandatory) – A unique identifier.
- **name** – A human readable name.
- **description** – A human readable description.
- **responsibleOrganization** – Information about the organization responsible for the creation/maintenance/administration of an object and its corresponding data.

6.2 ApplicationSchema

The information model representation of the application schema as defined in section 4.1.1.

6.2.1 Attributes

- **featureTypes** - A collection of names of the feature types that the application schema defines.

6.2.2 Relations

- **rule** – An application schema may be related to zero or more Rules.
- **ruleSet** – An application schema may be related to zero or more RuleSets.

6.3 StandardColor

A StandardColor is the definition of a color used to portray geographical data.

StandardColor instances represent named colors that can be used by Symbols in the Portrayal Registry. The intention is to define colors by hue rather than by intended usage, e.g. “green”, “red”, rather than “forest”, “building”. StandardColors are intended to encourage the use of standard colors in maps to e.g. reduce the cost of production of paper maps, or help increase readability of digital maps, by limiting the number of colors used.

6.3.1 Attributes

- **previewImage** – an image representing a preview of the StandardColor.

6.3.2 Relations

- **symbol** – A standardColor may be related to zero or more Symbols.

6.4 StandardFont

StandardFont instances represent type fonts for use in Symbols in the portrayal registry.

6.4.1 Attributes

- **previewImage** – an image representing a preview of the StandardFont.

6.4.2 Relations

- **symbol** – A StandardFont may be related to zero or more Symbols.

6.5 Symbol

The information model representation of the symbol concept as defined in section 4.1.17.

Symbols can be point symbols (e.g. images, SVG), area colors and patterns (e.g. image or other representation), line symbols or complex symbols. A symbol can also be empty when used for features in a data set that are not intended to be portrayed.

Symbols do not contain feature attribute dependencies, such as using different colors depending on the value of a specific attribute. These dependencies shall be implemented using Rules.

6.5.1 Attributes

- **previewImage** – An image representing a preview of the symbol.
- **nativeEncoding** – The encoding/format of the symbol data, expressed as a MIME type.
- **intendedUse** – A description of the intended use for the symbol.
- **areaOfApplication** – A description of the intended area of application, e.g. “Nautical Charts”.
- **symbolType** – Describes what type of geographical object this symbol is applicable to. Possible types are, among others; Point, Line, Polygon.

6.5.2 Relations

- **symbolSet** – A symbol may be related to zero or more SymbolSets.
- **standardFont** – A symbol may be related to zero or more StandardFonts.
- **standardColor** – A symbol may be related to zero or more StandardColors.
- **rule** – A symbol may be related to zero or more Rules.

6.6 SymbolSet

The information model representation of the symbol set concept as defined in section 4.1.18.

SymbolSets are used to group related Symbols. For instance, one SymbolSet might represent symbols from the S-52 standard for nautical chart symbology.

6.6.1 Attributes

- **supportedEncodings** – A list of the encodings/formats that the symbol set has been designed to support, expressed as MIME types.

6.6.2 Relations

- **symbol** (mandatory) – A SymbolSet shall be related to one or more Symbols.

6.7 Rule

The information model representation of the rule concept as defined in section 4.1.15.

Rules are typically intended to portray one or more types of geographic features. The featureTypes collection contains the name, or names, of these feature types.

Rule data may contain conditions on feature attributes, and in so doing be limited to an application schema where these attributes are defined. This limitation is defined by the ApplicationSchemas to which the Rule is related. If no application schema is defined, the rule is applicable to any application schema, meaning no attribute lookup is required to determine the visualization of features.

Rules do not contain visualization instructions – this is part of the associated Symbols.

6.7.1 Attributes

- **featureTypes** – A collection of feature type names to which the rule applies. Typically taken from the application schema. If no feature types are listed, the rule applies to any feature type.
- **nativeEncoding** – The encoding of the Rule data, expressed as a MIME type.

6.7.2 Relations

- **applicationSchema** – A Rule can be related to zero or more ApplicationSchema.
- **ruleSet** – A Rule can be related to zero or more RuleSets.
- **symbol** (mandatory) – A Rule must be related to at least one Symbol and can be related to several Symbols.

6.8 RuleSet

This is the information model representation of the rule set as defined in section 4.1.16.

RuleSet are used to group related Rules. A RuleSet shall contain only Rules that can be applied to the application schema specified by the applicationSchema relation, and should form a useful portrayal. A RuleSet may also contain any number of Rules which are not related to an application schema.

A RuleSet might e.g. represent a portrayal of an entire dataset such as VMap level 0 or Natural Earth, or a subset of a dataset, e.g. only natural features, or only roads and populated areas. The name and description atttributes should be set accordingly, so that consumers can understand the RuleSet's intended usage.

6.8.1 Attributes

- **supportedEncodings** – A list of the encodings/formats that the rule set has been designed to support.
- **previewImage** – an image representing a preview of the portrayal the RuleSet defines. This is typically a map image provided by the RuleSet creator.

6.8.2 Relations

- **applicationSchema** – A RuleSet may be related to zero or more ApplicationSchemas.
- **rule** (mandatory) – A RuleSet must be related to at least one Rule and is typically related to several Rules.

7. Required Data and Metadata Retrieval Operations

All interfaces fulfill a minimum set of data and metadata retrieval options, listed in this chapter. Since this section of the document is not an interface specification, no details such as return types or schemas are specified here. For such information, refer to either interface specification.

Please note that there might be minor differences, such as operation name or how many results are retrieved in one call, in the respective interfaces. There may also be more operations available than listed here, in either interface.

7.1 Metadata operations

This chapter lists the available metadata operations in all interfaces. It is up to the respective interface specifications to specify whether a parameter is optional or mandatory. Either interface may add to, but not subtract from, the parameter and operation list provided here. Each interface may also split these operations up into several smaller operations, but the sum of the available functionality in the operations of either interface must match, or exceed, the functionality of the operations listed in this chapter.

7.1.1 GetStandardColors

Parameters:

- **id** – The id of a specific StandardColor. If specified, only the matching StandardColor is returned.
- **name** - The name of the StandardColor.
- **keyword** – If this parameter is supplied all attributes of all StandardColors will be searched for the keyword and any matches will be returned.

Response: A collection of StandardColors matching the specified parameters.

7.1.2 GetStandardFonts

Parameters:

- **id** – The id of a specific StandardFont. If specified, only the matching StandardFont is returned.
- **name** - The name of the StandardFont.
- **keyword** – If this parameter is supplied all attributes of all StandardFonts will be searched for the keyword and any matches will be returned.

Response: A collection of StandardFonts matching the specified parameters.

7.1.3 GetSymbolSets

Parameters:

- **id** – The id of a specific SymbolSet. If specified, only the matching SymbolSet is returned.
- **name** - The name of the SymbolSet.
- **keyword** – If this parameter is supplied all attributes of all SymbolSets will be searched for the keyword and any matches will be returned.

Response: A collection of SymbolSets matching the specified parameters.

7.1.4 GetSymbols

Parameters:

- **id** - The id of a symbol. If specified, only the matching object is returned.

- **name** - The name of a symbol.
- **symbolSetId** - The id of a symbol set. If specified, only matching symbols from within this symbol set should be returned.
- **symbolType** - A reference to a classification node in the SymbolType classification scheme; the value may be a regular expression that matches multiple nodes. If specified, only symbols of the specified type are returned.
- **ruleId** - The id of a Rule in the registry. All Symbols referenced by that Rule should be returned, or no Symbols if no Rule with the given id exists.
- **keyword** - If this parameter is supplied all attributes of all Symbols will be searched for the keyword and any matches will be returned.

Response: A collection of Symbols matching the specified parameters.

7.1.5 GetRuleSets

Parameters:

- i**d** - The id of a RuleSet. If specified, only the matching object is returned.
- **name** - The name of a RuleSet.
- **applicationSchemaId** - The id of an application schema. Returned RuleSets must all be compliant with the identified application schema/feature catalogue.
- **keyword** - If this parameter is supplied all attributes of all RuleSets will be searched for the keyword and any matches will be returned.

Response: A collection of RuleSets matching the specified parameters.

7.1.6 GetRules

Parameters:

- **id** - The id of a Rule. If specified, only the matching object is returned.
- **name** - The name of a Rule.
- **ruleSetId** - The id of a symbol set. If specified, only matching symbols from within this symbol set should be returned.
- **featureType** - The name of a feature type, using only the local identifier, leaving out the application schema namespace part. If specified, the response is restricted to rules for only this feature type.
- **keyword** - If this parameter is supplied all attributes of all Rules will be searched for the keyword and any matches will be returned.

Response: A collection of Rules matching the specified parameters.

7.1.7 GetApplicationSchemas

Parameters:

- **id** - The id of an applicationSchema. If specified, only the matching object is returned.
- **name** - The name of an applicationSchema.
- **keyword** - If this parameter is supplied all attributes of all applicationSchema will be searched for the keyword and any matches will be returned.

Response: A collection of ApplicationSchemas matching the specified parameters.

7.2 Data operations

There is only one method of retrieving data. In this part of the specification the method will be referred to as “GetData”, but its name may be different in the different interface bindings. Each interface specification may contain extensions to this single data retrieval method, but all interface specifications must support this basic method.

7.2.1 GetData

Parameters:

- **id** – The unique identifier of the object. Mandatory.

Response: The data corresponding to the object. Depending on the interface used, the data may or may not be wrapped in an interface specific format.

8. Abstract test suite

8.1 Portrayal Registry Service

8.1.1 Operation response correctness

a) Test purpose: Verify that a portrayal registry satisfies all requirements of the GetStandardColors, GetStandardFonts, GetSymbolSets, GetSymbols, GetRuleSets, GetRules, GetApplicationSchemas and GetData operations.

b) Test Method: For each operation, make several requests using a variety of input parameters and combinations of input parameters. Verify that the objects returned from the operations correspond to the input parameters.

c) References: Respectively 7.1.1, 7.1.2, 7.1.3, 7.1.4, 7.1.5, 7.1.6, 7.1.7 and 7.2.1.

d) Test type: Basic

8.1.2 Information model completeness

a) Test purpose: Verify that the information model exposed by a portrayal registry satisfies the requirements expressed in this specification.

b) Test Method: Request at least one object of each class from the registry. Inspect the returned objects and verify that they as a minimum contain all mandatory attributes and relations as defined within this specification.

c) References: Respectively 6.2, 6.3, 6.4, 6.5, 6.6, 6.7 and 6.8.

d) Test type: Basic

8.1.3 Information model referential integrity

a) Test purpose: Verify that there are no invalid relations in the information model exposed by a portrayal registry.

b) Test Method: Request all objects from the registry. Verify that the objects that have relations defined are referencing objects that exist in the registry.

c) References: Respectively 6.2, 6.3, 6.4, 6.5, 6.6, 6.7 and 6.8.

d) Test type: Basic

9. Abstract use cases (informative)

This chapter lists some of the most likely use cases for a portrayal registry in an abstract fashion. That is, the uses cases contain no implementation-specific information, and can be viewed as abstract use cases, or conceptual use cases.

9.1 Get SymbolSet metadata

The client retrieves SymbolSet metadata by using the GetSymbolSets method specified in chapter 7.1.3. Since (in this example) the client specifies an identifier, SymbolSet metadata for only one SymbolSet is returned.

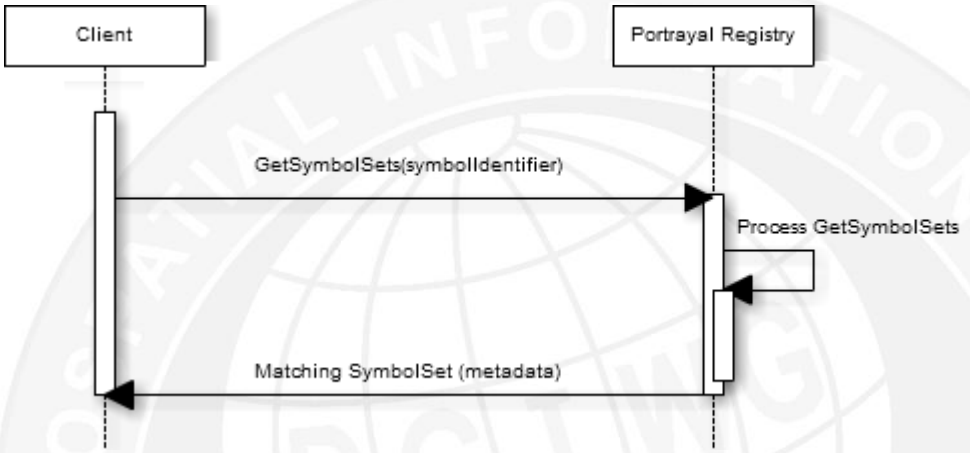


Figure 3: Retrieving SymbolSet metadata

9.2 Get Rule metadata for all rules applicable to feature type

The client retrieves metadata for all rules applicable to a certain featureType by performing a GetRules specified in chapter 7.1.6 and specifying the featureType as a parameter.

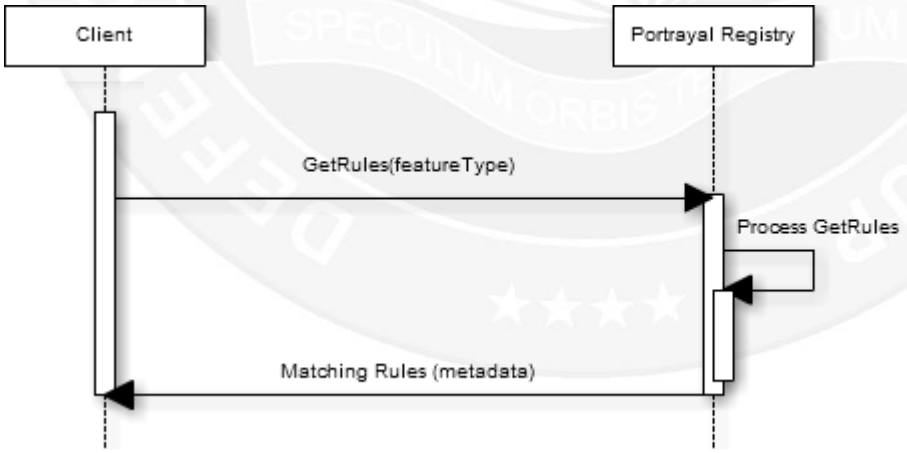


Figure 4: Retrieving Rule metadata for Rules applicable to featureType

9.3 Get Rule data for all rules in a RuleSet

The client retrieves Rule data for all rules in a RuleSet by following these steps (illustrated in Figure 5):

1. Retrieve RuleSet metadata. The client needs it to determine which Rules to retrieve.
2. For each Rule in the RuleSet:
    a. Retrieve the metadata of the Rule, the client needs it to determine which symbols to retrieve.
    b. Retrieve the Rule data

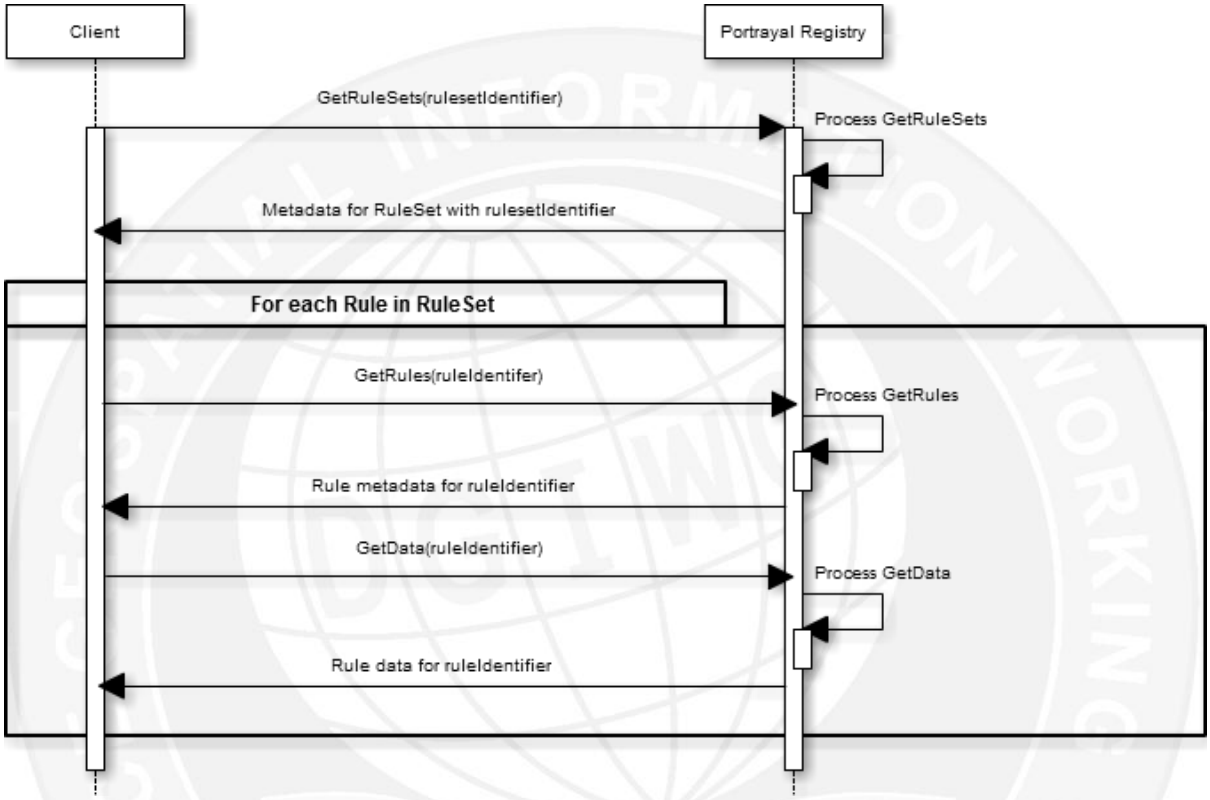


Figure 5: Retrieving Rule data for all Rules in a RuleSet

9.4 Using a Portrayal Registry with a Feature Portrayal Service

This use case illustrates the use of a portrayal registry (PRS) with a feature portryalal service (FPS). In this use case, many technical details have been removed for the purpose of understanding the relation between the FPS, PRS and client. For a more detailed, interface-specific version of this use case, see chapter B.5.7.

1. The client requests map images from the FPS and appends a portrayal registry reference when doing so.
2. The FPS extracts the PRS reference from the request and retrieves visualization data from the PRS.
3. The FPS then applies the visualization data when rendering map images and returns the map images.

This use case is illustrated in Figure 6: Using a PRS with an FPS.

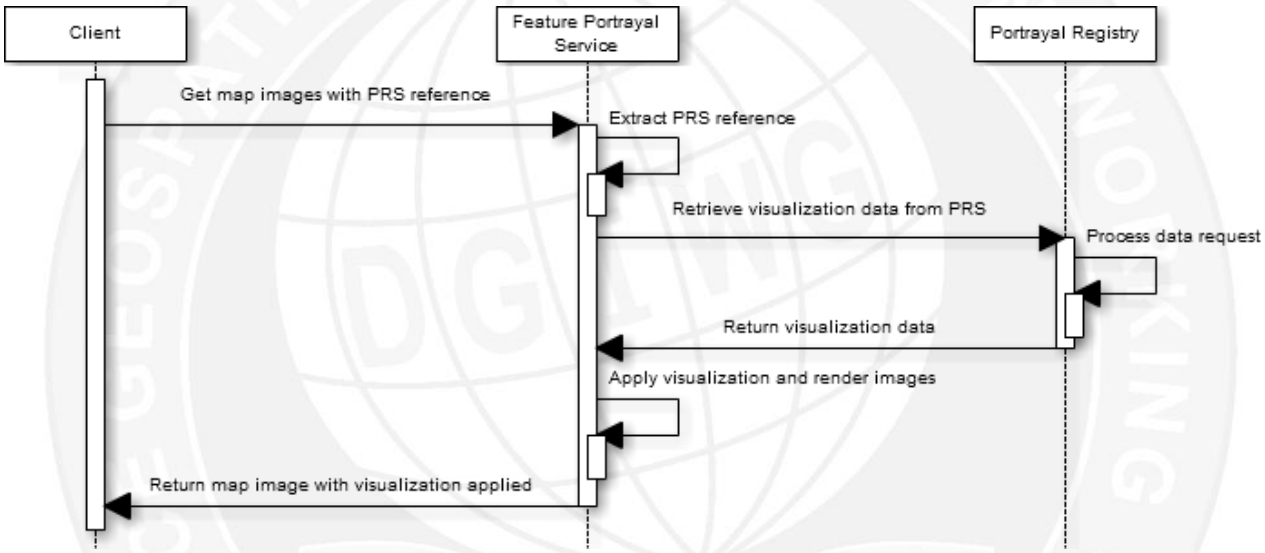


Figure 6: Using a PRS with an FPS

Annex A: CSW-ebRIM Interface Specification

(normative)

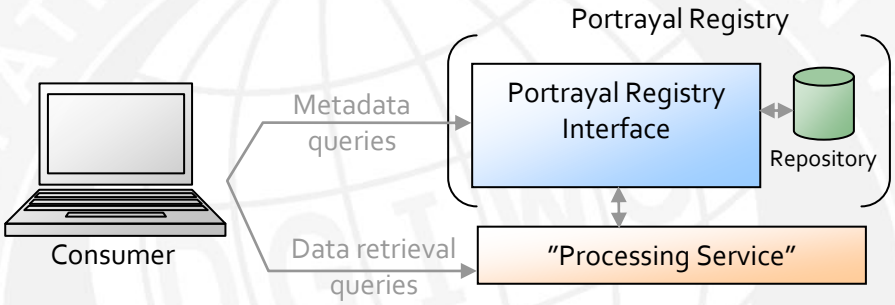
A.1 Scope and limitations

This document specifies a CSW-ebRIM Extension Package for Portrayal Registries, as one of the implementation options of the abstract concepts described in the body of this document.

A.2 CSW-ebRIM Service Interfaces

The following image shows the conceptual structure of a CSW ebRIM-based portrayal registry and its service interfaces.

Figure 7 – Portrayal registry services and interfaces conceptual structure



A.2.1 Portrayal registry interface

The *portrayal registry interface* is the service that presents the contents of the portrayal registry and allows users to interact with the registry. In this annex, the portrayal registry interface is often referred to simply as *the portrayal registry,* or *registry*.

The purpose of the portrayal registry interface is to retrieve *metadata*. The exception is the CSW-ebRIM *GetRepositoryItem* request, which returns the underlying *data*.

A.2.2 Vendor-specific "processing service"

This annex uses the term "Processing service" to conceptually describe the interface that portrayal consumers use to get the actual portrayal information (the *data*, or *repository items* in ebRIM terminology).

An important ability of the processing service is to return the portrayal information in several different formats – not only the native format of the respective items. A rule set might for example be returned both as a human-readable "symbol book" and as an SLD document.

Different implementations might implement this "processing service" in different ways – e.g. as an additional vendor-specific parameter to the CSW-ebRIM *GetRepositoryItem* request, or as a separate service interface.

A.3 CSW-ebRIM support

The Extension Package is based on the CSW-ebRIM Basic Extension Package [CSW-ebRIM-part2], which defines a basis for specialized registries based on the CSW profile of ebRIM [CSW-ebRIM-part1]. The CSW-ebRIM profile declares a set of operations, additional Slots on classes in ebRIM and provides the following extension points.

- New types of extrinsic objects and external links.
- New kinds of associations that link registry objects.
- Additional classification schemes—or classification nodes that augment an existing scheme—for classifying registry content.
- Predefined queries that reflect commonly executed metadata search patterns.
- Additional slots to further characterize particular types of registry objects.

Specifically, this does not allow a compliant specification to alter the registry service interface. It is not possible, for instance, to specify new *types of operations*, such as getting portrayal information in a specific encoding. It is only possible to specify new *predefined queries* (as defined in [CSW-ebRIM-part1], Clause 16), which are all invoked using an existing operation.
Implementations claiming conformance to this CSW-ebRIM interface specification shall be conformant to at least CSW-ebRIM Conformance Level 1, as per [CSW-ebRIM-part3]._

The Extension Package shall be represented as a rim:RegistryPackage instance. Package members are RegistryObjects that are subject to the constraint that a member object may only be deleted if the package as a whole is deleted. The rim:RegistryPackage is a formalized way to represent a Extension Package, through a set of elements and extensibility points offered by ebRIM that enable it to be tailored for specific purposes. The ‘id’ attribute of the rim:RegistryPackage instance shall be set to “urn:dgiwg:x-def:ebRIM-RegistryPackage:DGIWG:Portrayal” and the name attribute set to ‘Portrayal Registry extension package’ the rim:RegistryPackage instance shall be a member of the ‘root’ package, which contains all packages supported by the service, and shall contain the additional extrinsic objects, association types, classification schemes and nodes, slots, stored queries defined by the specification.

A.3.1 Data retrieval

There is only one operation in [CSW-ebRIM-part1] that allows retrieval of data, GetRepositoryItem. The GetRepositoryItem method has one parameter, the unique identifier of the extrinsic object representing the repository item to be retrieved.

The GetRepositoryItem request corresponds to the request described in chapter 7.2.1.

Refer to section A.2.2 for a discussion about how implementations might extend the functionality of the GetRepositoryItem method in a portrayal context.

A.4 Information model specification

A.4.1 Concepts from ebRIM model

This section briefly describes the data types, object types and object attributes that this extension package re-uses from the [ebRIM] model.

Conformance to this section is covered by the mandatory dependency to CSW-ebRIM Conformance Level 1 (cf. section A.3).

A.4.1.1 Data types

These data types are defined in [ebRIM].

String

"Used for unbounded Strings"

Set

"As defined by OCL: an unordered Collection in which an object can occur only once."

A.4.1.2 Object types

Defined in [CSW-ebRIM-part2]:

Image

"An Image item is a visual representation other than text. Both still and moving images fall under this category."

A.4.1.3 Object classes

The new classes in this Extension Package rely on attributes from the following [ebRIM] classes:

A.4.1.3.1 Identifiable

Identifiable.id:

"Each Identifiable instance MUST have a unique identifier which is used to refer to that object."

A.4.1.3.2 RegistryObject

RegistryObject.description:

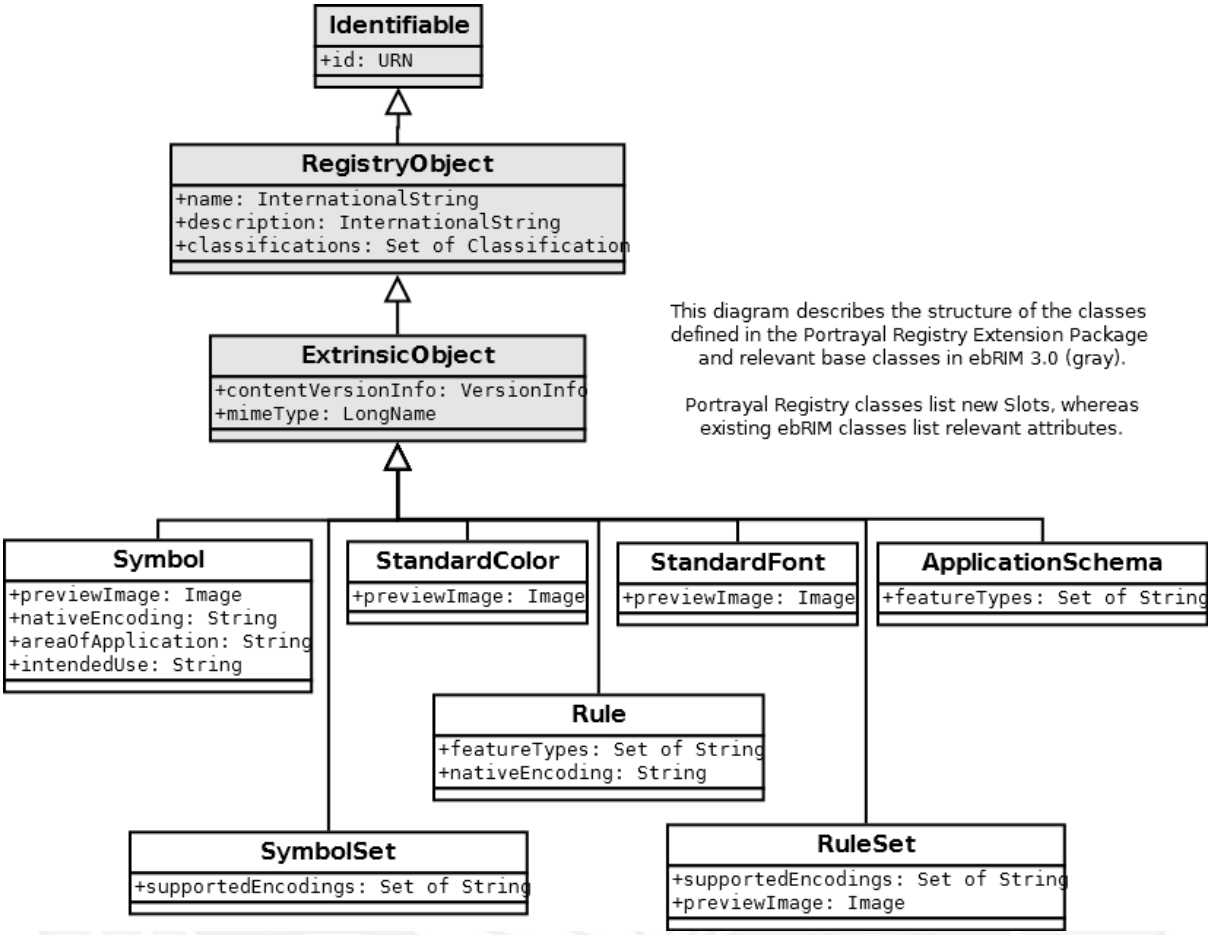
"Each RegistryObject instance MAY have textual description in a human readable and user-friendly form. This attribute is I18N capable and therefore of type InternationalString."

RegistryObject.name:

"Each RegistryObject instance MAY have a human readable name. The name does not need to be unique with respect to other RegistryObject instances. This attribute is I18N capable and therefore of type InternationalString."

A.4.2 New Extrinsic Objects defined in the Portrayal Registry Extension Package

Figure 8 – Class hierarchy for the Portrayal Registry Extension Package



This annex specifies a number of new extrinsic objects that shall be implemented by an implementation claiming conformance to this Extension Package.
Slots for the new classes are defined as either *mandatory* or *optional*. Mandatory Slots are core pieces of information, whereas optional Slots provide additional detail. Both mandatory and optional slots shall be implemented by a conformant implementation. Mandatory slots shall be instantiated in the medata (ebRIM instance) while optional ones may not.

A.4.2.1 Resource Identification

The objects defined in this annex shall be identified with Uniform Resource Names (URN), according to the DGIWG URN scheme [RFC6288].

The following URN structures are defined in the specification:

- *"urn:dgiwg:x-def:ebRIM-RegistryPackage:DGIWG:Portrayal"* is used as the value of the rim:RegistryPackage/@id attribute (cf. §A.3).
- "*urn:dgiwg:x-def:ebRIM-ObjectType:DGIWG*" shall be the root string for all Object Type URNs defined in this annex. The shortcut *{dgiwg-ebRIMObject}* is used in this annex.

- *“urn:dgiwg:x-def:ebRIM-AssociationType:DGIWG”* shall be the root string for all Association Type URNs defined in this annex. The shortcut *{dgiwg-ebRIMAssoType}* is used in this annex.
- ”*urn:dgiwg:x-def:ebRIM-ClassificationScheme:DGIWG*” shall be the root string for all Classification Scheme URNs defined in this annex. The shortcut *{dgiwg-ebRIMClassScheme}* is used in this annex.
- Classification nodes URNs are dependant on the name of the classification scheme to which they belong. They are specified in the chapter where the nodes are described.
- ”*urn:dgiwg:x-def:ebRIM-Query:DGIWG*” shall be the root string for all Classification Scheme URNs defined in this annex. The shortcut *{dgiwg-ebRIMQuery}* is used in this annex.

All shortcuts shall be replaced by the actual root string in the implementation.

NOTE: These URNs are tentative, as denoted by the ‘x-‘ prefix before ‘def’. They will be confirmed or updated once the DGIWG Naming Authority is in place.

A.4.2.1.1 StandardColor

A StandardColor object shall represent a standard color as defined in 6.3 - StandardColor.

Super classes: `ExtrinsicObject`
URN: `{dgiwg-ebRIMObject}:StandardColor`
The description attribute may contain a description of the colour (e.g. percentages of RGB or CMYK).

Table 2 – StandardColor slots

Slot name	Slot type	Description
`previewImage`	Image	*(optional)* A preview of the color.

A.4.2.1.2 StandardFont

A StandardFont object shall represent a standard font as defined in 6.4 - StandardFont.

Super classes: `ExtrinsicObject`
URN: `{dgiwg-ebRIMObject}:StandardFont`
The description attribute may contain a description of the intended use of the font.

Table 3 – StandardFont Slots

Slot name	Slot type	Description
`previewImage`	Image	*(optional)* A preview of the font.

A.4.2.1.3 Symbol

A Symbol object shall represent a symbol as defined in 6.5 - Symbol.

Super classes: `ExtrinsicObject`
URN: `{dgiwg-ebRIMObject}:Symbol`
The `supportedEncodings` Slot contains the names of the encodings into which the symbol may be translated. For example, a symbol defined in a specific subset of Symbology Encoding (bibliography [6]) might be possible to translate into KML (bibliography [7]). This Symbol should have `supportedEncodings` list both SE and KML, even though a given Processing service might not be able to perform the conversion.

Table 4 – Symbol Slots

Slot name	Slot type	Description
`previewImage`	Image	*(optional)* A preview of the symbol.
`nativeEncoding`	String	*(mandatory)* The encoding/format of the stored symbol representation, represented as a MIME type (e.g. image/gif, image/svg+xml).
`areaOfApplication`	String	*(optional)* A general description of the intended area of application, e.g. “Nautical charts”, “Air traffic control”, “Points of interest” …
`intendedUse`	String	*(optional)* A more detailed description of the intended use for the symbol. E.g. “North cardinal beacon, port-hand side”, “Restricted airspace, military area”, “Hospital” …

A.4.2.1.4 Rule

A Rule object shall represent a rule as defined in in 6.7 - Rule.

Super classes: `ExtrinsicObject`
URN: `{dgiwg-ebRIMObject}:Rule`
The `description` Slot may describe what the rule intends to achieve.

Table 5 – Rule Slots

Slot name	Slot type	Description
`featureTypes`	Set of String	*(optional)* The names of the feature types to which the rule applies. Typically taken from the application schema. If left empty, the rule applies to any feature type.
`nativeEncoding`	String	*(mandatory)* The encoding/format of the stored rule data, represented as a MIME type (e.g. application/vnd.ogc.se+xml).

A.4.2.1.5 ApplicationSchema

An ApplicationSchema object shall represent an application schema as defined in 6.2 - ApplicationSchema.
Super classes: `ExtrinsicObject`
URN: `{dgiwg-ebRIMObject}: ApplicationSchema`

Table 6 – ApplicationSchema Slots

Slot name	Slot type	Description
`featureTypes`	Set of String	*(optional)* A list of names of the feature types that the application schema defines.

A.4.2.1.6 SymbolSet

A SymbolSet shall represent a symbol set as defined in 6.6 - SymbolSet.

Super classes: `ExtrinsicObject`
URN: `{dgiwg-ebRIMObject}:SymbolSet`

Table 7 – SymbolSet Slots

Slot name	Slot type	Description
`supportedEncodings`	Set of String	*(mandatory)* List of encodings that the symbol set has been designed to support, represented as MIME types.

A.4.2.1.7 RuleSet

A RuleSet shall represent a rule set as defined in 6.8 - RuleSet.

Super classes: `ExtrinsicObject`
URN: `{dgiwg-ebRIMObject}:RuleSet`
The `supportedEncodings` Slot shall contain the names of all encodings that the content of the RuleSet are compatible with, even if any given Processing service might not be capable of performing a conversion between the encodings.

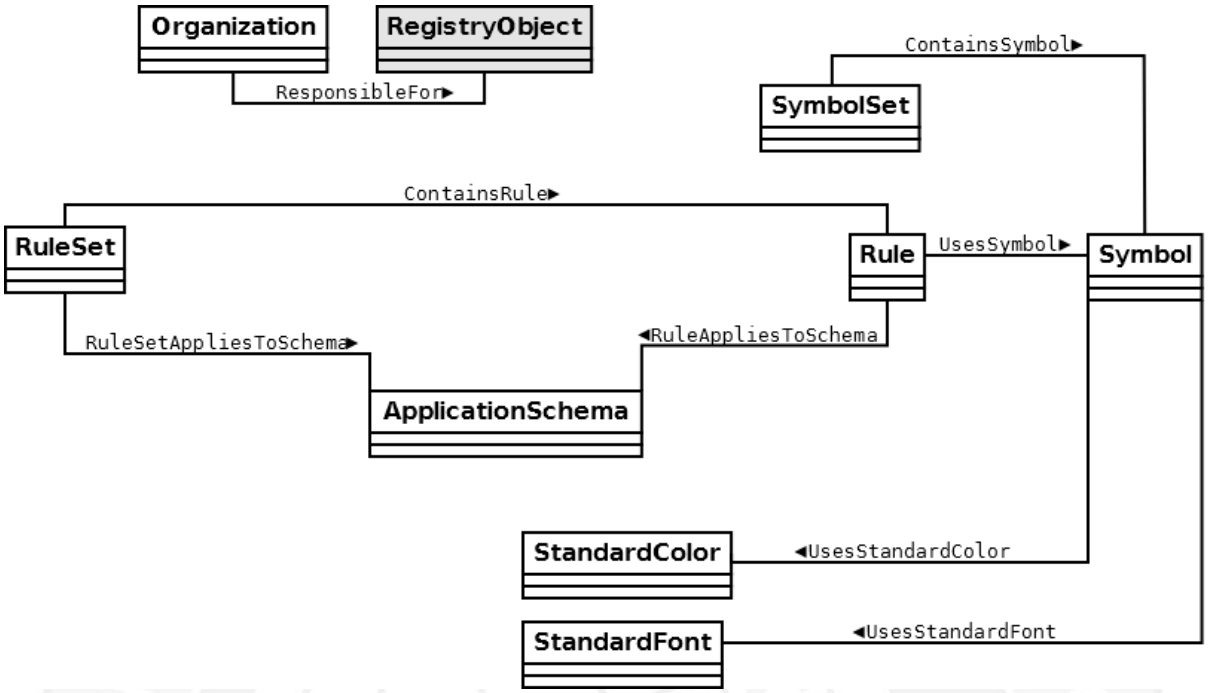
Table 8 – RuleSet Slots

Slot name	Slot type	Description
`previewImage`	Image	*(optional)* A preview of the portrayal the RuleSet defines. A map image uploaded by the owner of the RuleSet, meant to help consumers identify RuleSets suitable for their applications.
`supportedEncodings`	Set of String	*(optional)* List of encodings that the rule set has been designed to support, represented as MIME types.

A.4.3 Association Types

Associations are used to model relationships between RegistryObjects. Associations define two endpoints, a source and a target. This interface specification specifies a number of association types that shall be implemented by a conformant implementation.
Figure 9 illustrates the association types described in the following subchapters.

Figure 9 – Associations



A.4.3.1 ResponsibleFor

Defined in [ebRIM].
ResponsibleFor shall associate an Organization with at least one RegistryObject, specifying that the Organization is responsible for the linked RegistryObject's metadata and the corresponding repository item.

The primaryContact attribute of the Organization (of type Person) shall describe the official contact of an organization.

The instantiation of the ResponsibleFor association is optional.

A.4.3.2 ContainsSymbol

URN: {`dgiwg-ebRIMAssoType`}`:ContainsSymbol`
ContainsSymbol shall associate a SymbolSet with at least one Symbol, specifying that the Symbol is contained in the SymbolSet.

The instantiation of the ContainsSymbol association is mandatory for all SymbolSets in the registry.

A.4.3.3 ContainsRule

URN: {`dgiwg-ebRIMAssoType`}`:ContainsRule`
ContainsRule shall associate a RuleSet with at least one Rule, specifying that the Rule is contained in the RuleSet.

The instantiation of the ContainsRule association is mandatory for all RuleSets in the registry.

A.4.3.4 RuleSetAppliesToSchema

URN: {dgiwg-ebRIMAssoType}:RuleSetAppliesToSchema
RuleSetAppliesToSchema shall associate a RuleSet with an ApplicationSchema, specifying that the RuleSet is applicable to the referenced application schema.

The instantiation of the RuleSetAppliesToSchema association is optional.

A.4.3.5 UsesSymbol

URN: {dgiwg-ebRIMAssoType}:UsesSymbol
UsesSymbol shall associate a Rule with a Symbol, specifying that the Rule uses the Symbol.

The instantiation of the UsesSymbol association is mandatory for all Rules in the registry.

A.4.3.6 UsesStandardColor

URN: {dgiwg-ebRIMAssoType}:UsesStandardColor
UsesStandardColor shall associate a Symbol with a StandardColor, specifying that the Symbol uses the StandardColor.

The instantiation of the UsesStandardColor association is optional.

A.4.3.7 UsesStandardFont

URN: {dgiwg-ebRIMAssoType}:UsesStandardFont
UsesStandardFont shall associate a Symbol with a StandardFont, specifying that the Symbol uses the StandardFont.

The instantiation of the UsesStandardFont association is optional.

A.4.3.8 RuleAppliesToSchema

URN: {dgiwg-ebRIMAssoType}:RuleAppliesToSchema
RuleAppliesToSchema shall associate a Rule with an ApplicationSchema, specifying that the Rule is applicable to the associated application schema.

The instantiation of the RuleAppliesToSchema association is optional.

A.4.4 Additional ClassificationSchemes

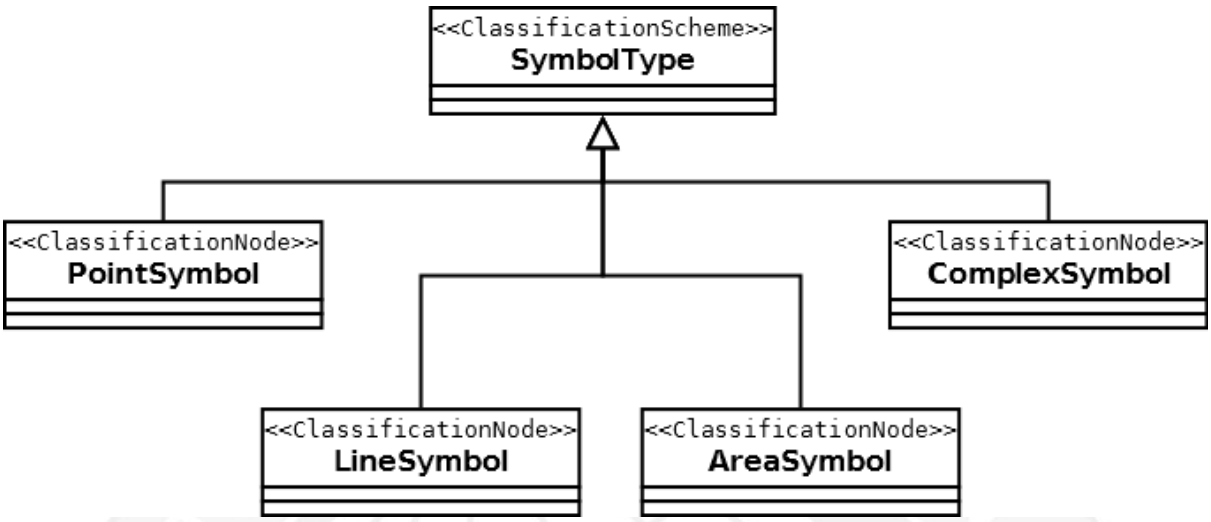
A ClassificationScheme describes a taxonomy, which is used to describe RegistryObjects. A ClassificationScheme consists of a tree of ClassificationNodes, each of which represents a category or family within the taxonomy. A RegistryObject may be classified by any number of ClassificationNodes from any number of ClassificationSchemes.
This interface specification defines an additional classification scheme that shall be implemented by implementations claiming conformance to this specification.

A.4.4.1 SymbolType ClassificationScheme

The SymbolType taxonomy, depicted in
Figure 10, determines the type of a Symbol instance. Setting the attribute ‘classifications’ of a Symbol instance, the symbol is classified as belonging to one, or more, of these categories. The classifications attribute is inherited from RegistryObject.

Symbol extrinsic objects shall instantiate the SymbolType classification.

Figure 10 – SymbolType taxonomy



SymbolType ClassificationSchema URN:
`{dgiwg-ebRIMClassScheme}:SymbolType`

A.4.4.1.1 PointSymbol

URN: `urn:dgiwg:x-symbolType:PointSymbol`
A PointSymbol represents one or more visualization instructions applicable to a point - images/icons or text rendering instructions.

A.4.4.1.2 LineSymbol

URN: `urn:dgiwg:x-symbolType:LineSymbol`
A LineSymbol represents one or more visualization instructions applicable to a curve geometry. These may be any combination of line styles (colors, patterns etc), points/icons that can be applied along the line in a given fashion and text rendering instructions.

A.4.4.1.3 AreaSymbol

URN: `urn:dgiwg:x-symbolType:AreaSymbol`
An AreaSymbol represent one or more visualization instructions applicable to a surface geometry. These may be any combination or area fill styles (colors, patterns etc.), line styles (typically applied along the outline of an area), points/icons (applied along the outline or distributed inside the area in a given fashion) and text rendering instructions.

A.4.4.1.4 ComplexSymbol

URN: `urn:dgiwg:x-symbolType:ComplexSymbol`
A complex symbol shall be a combination of visualization instructions which renders more complex geometries in a certain way. Typical instances of ComplexSymbol are tactical graphics, where multiple points or a line and an accompanying point are used together to define the final representation.

A.4.5 Predefined queries

The operations listed in chapter 7.1 - Metadata operations are implemented as predefined queries in this specification.

Predefined queries are invoked using the stored query framework defined in [CSW-ebRIM-part1], clause 16. The request is framed as a `<rim:adhocQuery>` element with Slots for any parameters to be invoked. This framework does not allow implementations to specify that some parameters are mandatory, but may instead return empty data lists if essential parameters are missing. The example request below invokes a CSW-ebRIM stored query to find all rule sets that use the keyword “LTDS” in any metadata field.

Example 1 – Invoking a predefined query using the CSW GetRecords context

```
<csw:GetRecords service="CSW" version="2.0.2" resultType="results"
                outputSchema="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0">
      <rim:AdhocQuery id=" urn:ogc:def:ebRIM-Query:OGC:Portrayal:getRuleSets">
            <rim:Slot name="keyword">
                  <rim:ValueList><rim:Value>*LTDS*</rim:Value></rim:ValueList>
            </rim:Slot>
      </rim:AdhocQuery>
</csw:GetRecords>
```

The following chapters list the predefined queries a Portrayal Registry shall implement to comply with this specification. Common to all of them is that if any optional parameter is not specified in a request, the corresponding filter criteria shall be ignored when performing the query. A parameter is considered to be ‘essential’ if it is required to perform a query. If an essential parameter is not supplied, the service shall respond with an exception report, explaining what caused the exception.

A.4.5.1 getStandardColors

Corresponds to the operation described in 7.1.1 - GetStandardColors.

Invoking the `getStandardColors` query returns a listing of StandardColors in the registry matching the query filter.

Table 9 – Predefined query: getStandardColors

Identifier	{dgiwg-ebRIMQuery}:getStandardColors
Name	getStandardColors
Parameters	`id` (optional) The id of a specific StandardColor. If specified, only the given color shall be returned. `name` (optional) The name of the StandardColor; the value may be a regular expression that matches multiple StandardColors. `keyword` (optional) If this parameter is supplied, all attributes and slots of all StandardColors, matching other search criteria, shall be searched for the keyword and any matches shall be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element corresponding to each StandardColor in the repository matching specified parameters.

A.4.5.2 getStandardFonts

Corresponds to the operation described in 7.1.2 - GetStandardFonts.

Invoking the `getStandardFonts` query returns a listing of standard font metadata from the registry matching the query filter.

Table 10 – Predefined query: getStandardFonts

Identifier	{dgiwg-ebRIMQuery}:getStandardFonts
Name	getStandardFonts
Parameters	`id` (optional) The id of a specific StandardFont. If specified, only the given font shall be returned. `name` (optional) The name of the font; the value may be a regular expression that matches multiple StandardFonts. `keyword` (optional) If this parameter is supplied, all attributes and slots of all StandardFonts, matching other search criteria, shall be searched for the keyword and any matches shall be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element corresponding to all StandardFonts in the repository matching specified parameters.

A.4.5.3 getSymbolSets

Corresponds to the operation described in 7.1.3 - GetSymbolSets.

Invoking the `getSymbolSets` query returns a listing of symbol set metadata from the registry matching the query filter.

Table 11 – Predefined query: getSymbolSets

Identifier	{dgiwg-ebRIMQuery}:getSymbolSets
Name	getSymbolSets
Parameters	`id` (optional) The id of a specific symbol set. If specified, only the given symbol set shall be returned. `logicalId` (optional) The `lid` of a specific symbol set. The `lid` attribute is unique to a linear succession of versions of a SymbolSet. If specified, the query should return all versions of a symbol set that match the other specified parameters. I.e., the `logicalId` attribute will have no effect if used in conjunction with the `id` attribute since the result returned when specifying the id attribute is always a subset of the result of a query performed specifying the logicalId. `name` (optional) The name of a symbol set; the value may be a regular expression that matches multiple symbol sets. `responsibleOrganizationId` (optional) The id of the organization responsible for the symbol set. If specified, only SymbolSets associated with the Organization with the given id by a ResponsibleFor Association should be returned. `keyword` (optional) If this parameter is supplied, all attributes and slots of all SymbolSets, matching other search criteria, will be searched for the keyword and any matches will be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element for each symbol set in the repository matching specified parameters.

A.4.5.4 getSymbols

Corresponds to the operation described in 7.1.4 - GetSymbols.

Invoking the `getSymbols` query returns a listing of symbol metadata from the registry matching the query filter.

Table 12 – Predefined query: getSymbols

Identifier	{dgiwg-ebRIMQuery}:getSymbols
Name	getSymbols
Parameters	`id` (optional) The id of a symbol. `name` (optional) The name of a symbol; the value may be a regular expression that matches multiple symbols. `symbolSetId` (optional) The id of a symbol set. If specified, only matching symbols from within this symbol set shall be returned. `symbolType` (optional) A reference to a classification node in the SymbolType classification scheme; the value may be a regular expression that matches multiple nodes. If specified, only symbols of the specified type shall be returned. `supportedEncoding` (optional) The uniquely identifying name (currently implementation-specific) of an encoding that all returned symbols shall support. `keyword` (optional) If this parameter is supplied, all attributes and slots of all Symbols, matching other search criteria, shall be searched for the keyword and any matches shall be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element for each matching symbol in the symbol set.

A.4.5.5 getRuleSets

Corresponds to the operation described in 7.1.5 - GetRuleSets.

Invoking the `getRuleSets` query returns a listing of rule set metadata from the registry matching the query filter.

Table 13 – Predefined query: getRuleSets

Identifier	{dgiwg-ebRIMQuery}:getRuleSets
Name	getRuleSets
Parameters	`id` (optional) The id of a specific rule set. If specified, only the given rule set shall be returned. `logicalId` (optional) The `lid` of a specific rule set. The `lid` attribute is unique to a linear succession of versions of a RuleSet. If specified, the query should return all versions of a rule set that match the other specified parameters. I.e., the `logicalId` attribute will have no effect if used in conjunction with the `id` attribute since the result returned when specifying the id attribute is always a subset of the result of a query performed specifying the logicalId. `name` (optional) The name of a rule set; the value may be a regular expression that matches multiple rule sets. `applicationSchemaId` (optional) The id of an application schema. Returned rule sets shall all be compliant with the identified application schema/feature catalogue. `responsibleOrganizationId` (optional) The id of the organization responsible for the rule set. `supportedEncoding` (optional) The uniquely identifying name (currently implementation-specific) of an encoding that all returned rule sets must support. `keyword` (optional) If this parameter is supplied, all attributes and slots of all RuleSets, matching other search criteria, shall be searched for the keyword and any matches shall be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element for each rule set in the repository matching specified parameters.

A.4.5.6 getRules

Corresponds to the operation described in 7.1.6 - GetRules.

Invoking the `getRules` query returns a listing of rule metadata from the registry matching the query filter.

Table 14 – Predefined query: getRules

Identifier	{dgiwg-ebRIMQuery}:getRules
Name	getRules
Parameters	`id` (optional) The id of a specific rule. If specified, only the given rule shall be returned. `name` (optional) The name of a rule; the value may be a regular expression that matches multiple rules. `ruleSetId` (optional) The id of a rule set. If specified, the response shall be restricted to this RuleSet. `featureType` (optional) The name of a feature type. If specified, the response shall be restricted to rules for only this feature type. `keyword` (optional) If this parameter is supplied, all attributes and slots of Rules, matching other search criteria, shall be searched for the keyword and any matches shall be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element for each rule in the rule set.

A.4.5.7 getApplicationSchemas

Corresponds to the operation described in 7.1.7 - GetApplicationSchemas.

Invoking the `getApplicationSchemas` query returns a listing of application schema metadata from the registry matching the query filter.

Table 15 – Predefined query: getApplicationSchemas

Identifier	{dgiwg-ebRIMQuery}:getApplicationSchemas
Name	getApplicationSchemas
Parameters	`id` (optional) The id of an application schema. `name` (optional) The name of an application schema; the value may be a regular expression that matches multiple application schemas. `keyword` (optional) If this parameter is supplied, all attributes and slots of all ApplicationSchemas, matching other search criteria, shall be searched for the keyword and any matches shall be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element for each matching application schema in the registry.

A.4.5.8 getRulesUsingSymbol

Invoking the query returns.
Invoking the `getRulesUsingSymbol` query returns all Rules in the registry that reference the specified Symbol.

Table 16: Predefined query: getRulesUsingSymbol

Identifier	{dgiwg-ebRIMQuery}:getRulesUsingSymbol
Name	getRulesUsingSymbol
Parameters	`id` (optional) The id of a Symbol. Returned Rules shall reference this Symbol. `logicalId` (optional) The `lid` of a specific Symbol. The `lid` attribute is unique to a linear succession of versions of a Symbol. If specified, the query shall return all Rules using any version of the specified logicalId. I.e., the `logicalId` attribute will have no effect if used in conjunction with the `id` attribute since the result returned when specifying the id attribute is always a subset of the result of a query performed specifying the logicalId. Note: if no parameter is specified, an empty response will be returned.
Response	A `<csw:GetRecordsResponse>` element containing a `<rim:ExtrinsicObject>` element for each matching RuleSet referencing the Symbol, or a specific version of the Symbol.

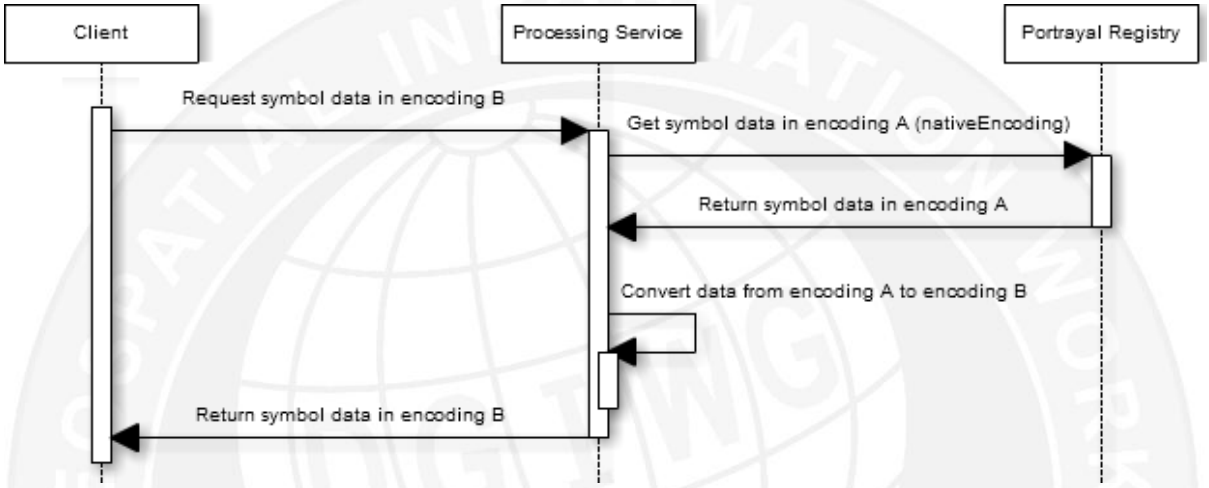
A.5 Intended use cases

In this chapter the primary intended use cases for the portrayal registry, and the processing service, are described.

A.5.1 Retrieving symbol data

A consumer uses the Processing service to retrieve the portrayal data for a symbol with a unique id in a specific encoding. The encoding is different that the nativeEncoding of the Symbol and therefore a conversion must be performed by the processing service.

Figure 11 – Sequence diagram for symbol data retrieval use case



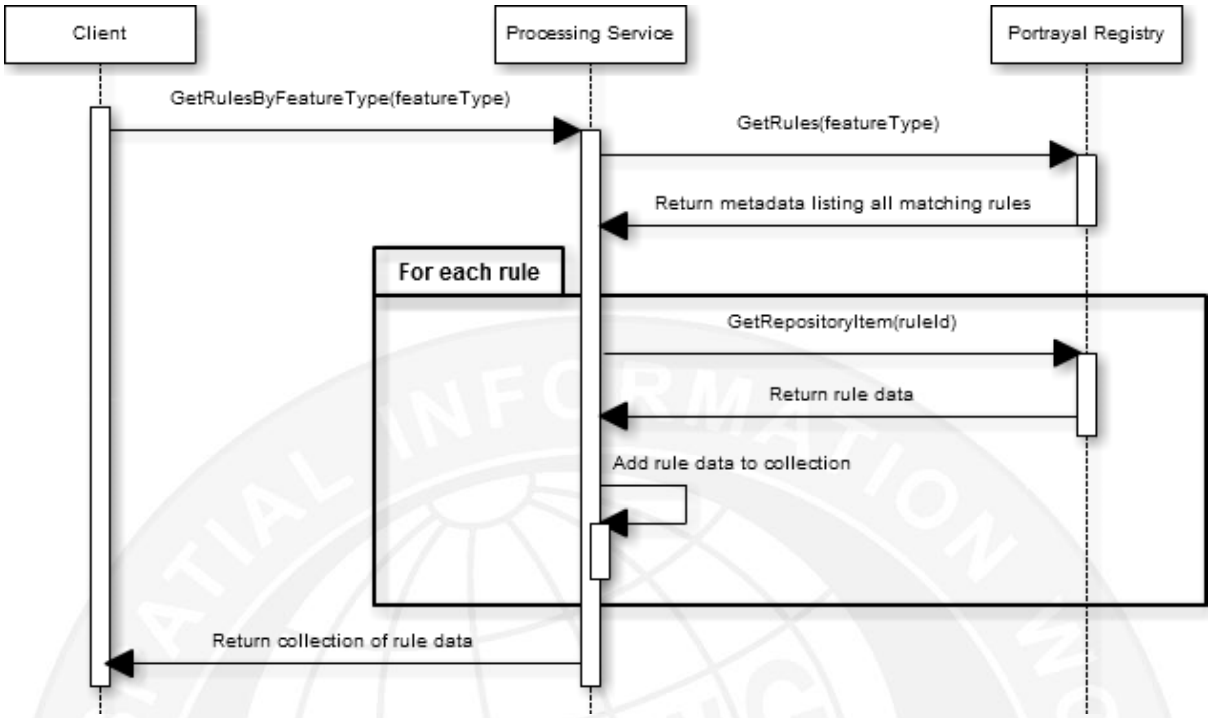
This use case uses the GetRepositoryItem request, described in chapter A.3.1.

A.5.2 Retrieving Rules applicable to a certain FeatureType

A consumer provides the processing service with a FeatureType and a RuleSetId. The Processing service then retrieves metadata describing all Rules from the referenced RuleSet which are applicable to the provided FeatureType. The retrieved metadata is used, by the processing service, to retrieve Rule data for each matching Rule. The Rule data is collected and finally delivered to the Consumer.

If one wishes to retrieve only the metadata, not the actual data, for each Rule matching the specified criteria steps 2 and 3 in Figure 12 need to be performed. These two steps do not have to be performed by the processing service but can just as well be performed by the Consumer.

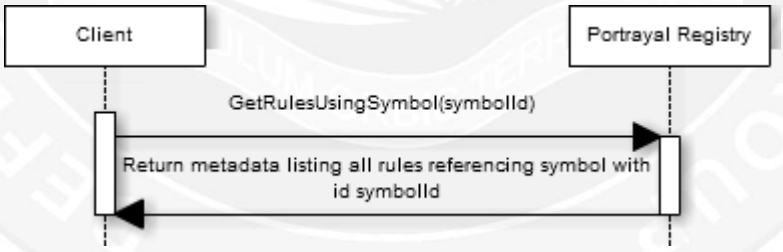
Figure 12 – Sequence diagram for FeatureType-specific Rule retrieval use case



This use case utilizes the getRules query, defined in chapter A.4.5.6, and the GetRepositoryItem method.
It is also possible to retrieve all Rules, in any RuleSet, applicable to a specific FeatureType by repeating this use case and not specifying a RuleSetId.

A.5.3 Retrieving metadata for all Rules referencing a Symbol

A consumer provides a SymbolId and performs the predefined getRulesUsingSymbol query. The response will contain metadata describing all Rules referencing the provided SymbolId.

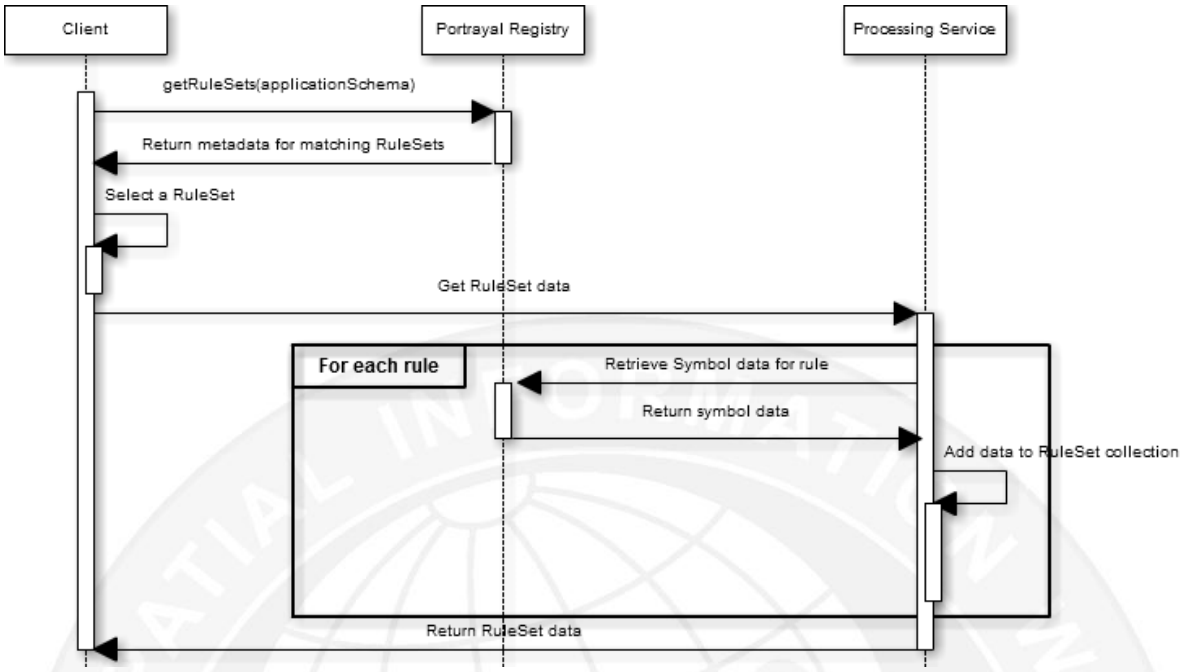


This use case uses the predefined query getRulesUsingSymbol, described in chapter A.4.5.8.

A.5.4 Retrieving RuleSet Data

A consumer uses the Portrayal Registry to find a rule set suitable for its geo data. The consumer then contacts the Processing Service to get the portrayal information in the desired encoding.

Figure 13 – Sequence diagram for consuming portrayal information use case



This use case uses the predefined getRuleSets query to retrieve RuleSet metadata and the getRules query to retrieve Rule metadata. For data retrieval, GetRepositoryItem is used by the processing service for Symbols and for all items referenced by Symbols.

A.6 CSW-ebRIM Interface Abstract Test Suite

A.6.1 CSW-ebRIM support

a) Test purpose: Check that the implementation under test is a valid implementation of Conformance Level 1 of CSW-ebRIM 1.0.1
b) Test method: Pass the tests for Conformance Level 1 of CSW-ebRIM 1.0.1. This includes support of the Basic extension package.
c) References: A.3
d) Test type: Capability

A.6.2 Portrayal Registry extension package availability

a) Test purpose: Check that the Portrayal Registry extension package is available as a supported extension package and is complete.
b) Test method: Verify that a rim:RegistryPackage element with id attribute set to “urn:ogc:def:ebRIM-package:OGC:Portrayal” and Name property set to ‘Portrayal Registry extension package’ is a member of the ‘root’ package, which contains all packages supported by the service, and contains the additional extrinsic objects, association types, classification schemes and nodes, slots, stored queries defined by the specification. Pass if the assertion is satisfied; fail otherwise.
c) References: A.3
d) Test type: Capability

A.6.3 Portrayal Registry extension package implementation

a) Test purpose: Check that all extrinsic objets, association types, classification schemes and stored queries of the Portrayal Registry extension package are correctly implemented.
b) Test method: Using GetRecords calls and an instance of the complete ebRIM information model, verify that the implementation under test supports:
   - Symbol, SymbolSet, Rule, RuleSet, StandardFont, StandardColor and ApplicationSchema extrinsic objects and all of their mandatory and optional slots;
   - SubmitterOf, ResponsibleFor, ContainsSymbol, ContainsRule, RuleSetAppliesToSchema, UsesSymbol, UsesStandardColor, UsesStandardFont, RuleAppliesToSchema association types.
   - SymbolType classification scheme,
   - getStandardColors, getStandardFonts, getSymbolSets, getSymbols, getRuleSets, getRules and getApplicationSchemas stored queries.

c) References: A.4, 8.1.1
d) Test type: Capability

A.6.4 ebRIM metadata completeness

a) Test purpose: Check that the content of the ebRIM registry is a correct instantiation of the extension package.
b) Test method: Verify that a representative sample of ebRIM metadata instantiates at least one extrinsic object with all mandatory slots, classification scheme and association types according to specification.
c) References: A.4, 8.1.2
d) Test type: Capability

A.6.5 ebRIM metadata referential integrity

a) Test purpose: Verify that there are no invalid relations in the ebRIM metadata.
b) Test Method: Request all association objects from the ebRIM registry. Verify that the source and target objects exist in the registry.
c) References: 0, 8.1.3
d) Test type: Capability

Annex B: RESTful Interface Specification

(normative)

B.1 Scope and limitations

This document specifies a RESTful interface for Portrayal Registries, as one of the implementation options of the abstract concepts described in the body of this document.

B.2 General design principles

This chapter outlines the general design principles of the REST interface described in this annex. The purpose is to make it easier to understand the upcoming chapters.

B.2.1 Top-level resources

The REST interface consists of three top-level resources:

- **metadata**, which is used to retrieve metadata for individual resources in the registry. It has subresources for each type of object in the information model. The metadata interface is described in detail in chapter B.4.5.

  Example: *http://www.example.com/portrayalregistry/**metadata/rule/ABC123.json*** returns metadata for rule ‘ABC123’ in JSON format.

- **data**, which is used to retrieve the underlying portrayal data for resources in the registry. The data interface is described in detail in chapter B.4.7.

  Example: *http://www.example.com/portrayalregistry/**data/ABC123*** returns data for the object ‘ABC123’ in its native encoding.

- **search**, which is used to return metadata for resources in the registry matching a specified search criteria. The search interface is described in detail in chapter B.4.6.

  Example: *http://www.example.com/portrayalregistry/**search/rules.xml?name=river*** returns metadata for all rules whose name contains ‘river’ in XML format.

B.2.2 Return types

Metadata and search requests can return either [XML], JSON (cf. [RFC6570]) or [HTML] formatted metadata depending on a format extension or content-type parameter. For more information refer to chapter B.4.2.

Data requests return the portrayal data in its native encoding or an encoding specified in the content-type parameter.

B.2.3 Capabilities documents

It is possible to retrieve capabilities documents for all three top-level resources by accessing their respective **capabilities** subresources.

Example: *http://www.example.com/portrayalregistry/**metadata/capabilities.json*** returns capabilities for the metadata interface.

For details on the capabilities requests, refer to sections B.4.5.8, B.4.6.8 and B.4.7.2.

B.3 Data types

This chapter contains a specification for all data types that are part of the REST interface. Chapter B.4 explains how these data types are used in the interface.

Where applicable, example objects are included both for the [XML] encoding specified in chapter B.6 and the JSON [RFC6570] encoding specified in chapter B.7.

These two encodings, as well as human-readable [HTML] encoding, are mandatory for all registry implementations – refer to chapter B.4.2 for details.

B.3.1 Information Model Types

This chapter specifies all objects that correspond to the information model specified in chapter 6
.

B.3.1.1 ItemBase

This object is the base for all representations of metadata. Unless specified otherwise, all Object Types specified in section B.3.1 extend this object. This object is not exposed to end users, therefore no examples are provided specifically for this object.

Table 17: ItemBase Fields

Field	Type	Description
Identifier	string	A unique identifier for this object
Name	string	A human readable name.
Description	string	A human readable description of the object and/or its contents.
ResponsibleOrganization	string	The organization responsible for the creation and/or maintenance of the object.

B.3.1.2 SymbolClassification

This object does not extend the ItemBase object. This object provides a means to classify symbols as either “complex”, “line”, “polygon” or “point”.

This object is not exposed to end users in any other form than as a subset of a Symbol object, therefore no examples are provided specifically for this object; see 0 for an example in which this object is used.

B.3.1.3 Symbol

Corresponds to the object described in 6.5 - Symbol.

The Symbol object contains the metadata of a symbol.

Table 18: Symbol Fields

Field	Type	Description
PreviewImage	Link	A Link, containing a URL for the preview image.
NativeEncoding	string	The native encoding of this symbol, expressed as a MIME type.
IntendedUse	string	The intended use of this symbol.
AreaOfApplication	string	The area of application for this symbol, e.g. “tactical” or “nautical”.
SymbolType	SymbolClassification (string)	Describes what type of geographical data this symbol is intended to visualize.
SymbolSetReferences	SymbolSetReferences	A list of Links representing symbolSets in which this symbol is used.
RuleReferences	RuleReferences	A list of Links representing rules in which this symbol is used.
StandardFontReferences	StandardFontReferences	A list of Links representing standardFonts which this symbol uses.
StandardColorReferences	StandardColorReferences	A list of Links representing standardColors which this symbol uses.

To view the XML schema for this object, refer to chapter B.6.

B.3.1.3.1 XML example object

```
<?xml version="1.0"?>
<Symbol xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Identifier>2324-7828-9342</Identifier>
  <Name>Enhanced coastal lines</Name>
  <Description>This symbol visualizes coastal lines.</Description>
  <PreviewImage xlink:href="http://www.prs.net/data/3213-5252-9494.png" />
  <NativeEncoding>application/vnd.ogc.se+xml;version=1.1</NativeEncoding>
  <AreaOfApplication>Nautical</AreaOfApplication>
  <SymbolType>Line</SymbolType>
  <SymbolSetReferences />
  <RuleReferences />
  <StandardFontReferences />
  <StandardColorReferences>
    <Link xlink:href="http://www.prs.net/metadata/standardColor/9348-7575-3223.xml" />
    <Link xlink:href="http://www.prs.net/metadata/standardColor/9378-7588-3223.xml" />
  </StandardColorReferences>
</Symbol>
```

B.3.1.3.2 JS Example object

```
{
    identifier: "ee43-aa54",
    name: "symbol example",
    description: "An example symbol object",
    responsibleOrganization: "Carmenta AB",
    nativeEncoding:"application/vnd.google-earth.kml+xml",
    areaOfApplication:"nautical charts",
    intendedUse:"boats",
    symbolClassification:"Complex",
    previewImage:"http://www.images.com/previewImageFont.jpg",
    symbolSetReferences:["http://www.prs.net/metadata/symbolSets/9191-9342.xml"],
    ruleReferences:["http://www.prs.net/metadata/rules/2324-1122.xml"],
    standardFontReferences:[],
    standardColorReferences:["http://www.prs.net/metadata/standardColors/5678-9342.xml"]
}
```

B.3.1.4 SymbolSet

Corresponds to the object described in 6.6 - SymbolSet.

The SymbolSet object contains the metadata of a SymbolSet.

Table 19: SymbolSet Fields

Field	Type	Description
SymbolReferences	SymbolReferences	A collection of Links, representing the Symbols used by this SymbolSet
SupportedEncodings	string	Comma-separated list of MIME types that this symbol set is designed to be encoded in.

To view the XML schema for this object, refer to chapter B.6.

B.3.1.4.1 XML Example object

```
<?xml version="1.0"?>
<SymbolSet xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Identifier>27832378-892389238932</Identifier>
  <Name>Nautical symbols</Name>
  <Description>Symbols suitable for nautical charts</Description>
  <ResponsibleOrganization>Acme</ResponsibleOrganization>
  <SymbolReferences>
    <Link xlink:type="simple" xlink:href="http://www.prs.net/metadata/symbol/2324-7828-
9342.xml" />
  </SymbolReferences>
</SymbolSet>
```

B.3.1.4.2 JS Example object

```
{
    identifier: "a3434-eb767",
    name: "symbolSet",
    description: "An example symbolSet object",
    responsibleOrganization: "Carmenta AB",
    symbolReferences:["http://www.prs.net/metadata/symbols/9889-9342.xml"]
}
```

B.3.1.5 Rule

Corresponds to the object described in 6.7 - Rule.

The Rule object contains the metadata of a Rule.

Table 20: Rule Fields

Field	Type	Description
NativeEncoding	string	The native encoding of this symbol, expressed as a MIME type.
FeatureTypes	Collection of strings	A collection of feature type names.
SymbolReferences	SymbolReferences	A collection of Links.
ApplicationSchemaReferences	ApplicationSchemaReferences	A collection of Links.
RuleSetReferences	RuleSetReferences	A collection of Links.

To view the XML schema for this object, refer to chapter B.6.

B.3.1.5.1 XML Example object

```
<?xml version="1.0" ?>
 <Rule xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Identifier>923480230423098</Identifier>
  <Name>myRule</Name>
  <Description>Test rule</Description>
  <NativeEncoding>application/vnd.ogc.se+xml; version=1.1</NativeEncoding>
   <FeatureTypes>
    <FeatureType>FeatureTypeZeta</FeatureType>
  </FeatureTypes>
  <SymbolReferences>
    <Link xlink:href="http://www.prs.net/metadata/symbol/3242542.xml" />
    <Link xlink:href="http://www.prs.net/metadata/symbol/3266254.xml" />
    <Link xlink:href="http://www.prs.net/metadata/symbol/3277754.xml" />
  </SymbolReferences>
   <ApplicationSchemaReferences>
    <Link xlink:href="http://www.prs.net/metadata/applicationSchema/324234.xml" />
    <Link xlink:href="http://www.prs.net/metadata/applicationSchema/399274.xml" />
    <Link xlink:href="http://www.prs.net/metadata/applicationSchema/377254.xml" />
  </ApplicationSchemaReferences>
  <RuleSetReferences />
</Rule>
```

B.3.1.5.2 JS Example object

```
{
    identifier: "234234-364364",
    name: "anyName",
    description: "An example rule object",
    responsibleOrganization: "Carmenta AB",
    nativeEncoding:"application/vnd.google-earth.kml+xml",
       featureTypes:["BuiltUpArea"],
       symbolReferences:["http://www.restprs.net/symbols/34543-23423"],
       applicationSchemaReferences:["http://www.restprs.net/applicationSchemas/ee664-
23423"],
       ruleSetReferences:["http://www.restprs.net/ruleSets/aaeefa-23423"]
}
```

B.3.1.6 RuleSet

Corresponds to the object described in 6.8 - RuleSet.

The RuleSet object contains the metadata of a RuleSet.

Table 21: RuleSet Fields

Field	Type	Description
PreviewImage	Link	A Link pointing to a preview image depicting a visualization using this rule set.
SupportedEncodings	string	Comma-separated list of MIME types that this rule set is designed to be encoded in.
RuleReferences	RuleReferences	A collection of Links. Each Link represents a URL linking to the metadata object of the Rule.

To view the XML schema for this object, refer to chapter B.6.

B.3.1.6.1 XML Example object

```
<?xml version="1.0" ?>
 <RuleSet xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Identifier>6462-3166-9246</Identifier>
  <Name>Nautical Rules</Name>
  <Description>Rules for nautical charts</Description>
  <ResponsibleOrganization>Seafarers club</ResponsibleOrganization>
  <PreviewImage />
   <RuleReferences>
    <Link xlink:href="http://www.prs.net/metadata/rule/923480230423098.xml" />
  </RuleReferences>
   <ApplicationSchemaReferences>
    <Link xlink:href="http://www.prs.net/metadata/applicationSchema/89232323-
123231312.xml" />
  </ApplicationSchemaReferences>
</RuleSet>
```

B.3.1.6.2 JS Example object

```
{
    identifier: "1765-9355",
    name: "baseName",
    description: "An example ruleSet object",
    responsibleOrganization: "Carmenta AB",
    previewImage:"http://www.images.com/previewImage.jpg",
    ruleReferences:["http://www.restprs.net/metadata/rules/9823-
7845.xml","http://www.restprs.net/metadata/rules/9823-2323.xml"]

}
```

B.3.1.7 ApplicationSchema

Corresponds to the object described in 6.2 - ApplicationSchema.

The ApplicationSchema object contains the metadata of an ApplicationSchema.

Table 22: ApplicationSchema Fields

Field	Type	Description
FeatureTypes	Collection of strings	A collection of feature type names.
RuleReferences	RuleReferences	A collection of Links.
RuleSetReferences	RuleSetReferences	A collection of Links.

To view the XML schema for this object, refer to chapter B.6.

B.3.1.7.1 XML Example object

```
<?xml version="1.0"?>
<ApplicationSchema xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Identifier>2389-9842-7835</Identifier>
  <Name>SomeSchema</Name>
  <Description>This schema describes the tactical standard 2566B</Description>
  <ResponsibleOrganization>Appschemas incorporated</ResponsibleOrganization>
  <FeatureTypes>
    <FeatureType>FTAlpha</FeatureType>
    <FeatureType>FTBeta</FeatureType>
  </FeatureTypes>
  <RuleReferences />
  <RuleSetReferences />
</ApplicationSchema>
```

B.3.1.7.2 JS Example object

```
{
    identifier: "1234123-123123123",
    name: "appSchemaName",
    description: "An example applicationSchema object",
    responsibleOrganization: "Carmenta AB",
    ruleReferences: [],
    ruleSetReferences: [],
    featureTypes: ["BuiltUpArea", "DestroyedBuiltUpArea"]
}
```

B.3.1.8 StandardFont

Corresponds to the object described in 6.4 - StandardFont.

The StandardFont object contains the metadata of a StandardFont.

Table 23: StandardFont Fields

Field	Type	Description
PreviewImage	Link	A Link to a preview image depicting a visualization using this StandardFont.
SymbolReferences	SymbolReferences	A collection of Links. A Link represents a URL linking to the metadata object of the Symbol.

To view the XML schema for this object, refer to chapter B.6.

B.3.1.8.1 XML Example object

```
<?xml version="1.0" ?>
 <StandardFont xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Identifier>KSJA-52532-SDOk</Identifier>
  <Name>Highways</Name>
  <Description>Font to be used for printing highway names</Description>
  <ResponsibleOrganization>Traffic works</ResponsibleOrganization>
  <PreviewImage xlink:href="http://www.prs.net/data/3213-6632-9494.png" />
   <SymbolReferences>
    <Link xlink:href="http://www.prs.net/metadata/symbol/2324-7828-9342.xml" />
    <Link xlink:href="http://www.prs.net/metadata/symbol/9232-5353-9392.xml" />
  </SymbolReferences>
</StandardFont>
```

B.3.1.8.2 JS Example object

```
{
    identifier: "8934-2323",
    name: "standardFont 89B",
    description: "An example standardFont object",
    responsibleOrganization: "Carmenta AB",
    previewImage:"http://www.images.com/previewImageFont.jpg",
    symbolReferences:["http://www.prs.net/metadata/symbols/2324-9342.xml"]
}
```

B.3.1.9 StandardColor

Corresponds to the object described in 6.3 - StandardColor.

The StandardColor object contains the metadata of a StandardColor.

Table 24: StandardColor Fields

Field	Type	Description
PreviewImage	Link	A link to a preview image depicting a visualization using this StandardColor.
SymbolReferences	SymbolReferences	A collection of Links. A Link represents a URL linking to the metadata object of the Symbol.

To view the XML schema for this object, refer to chapter B.6.

B.3.1.9.1 XML Example object

```
<?xml version="1.0" ?>
 <StandardColor xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Identifier>LASD-9283-asWE</Identifier>
  <Name>LakesAndRivers 66</Name>
  <Description>Great color to visualize lakes and rivers</Description>
  <ResponsibleOrganization>Nautical chart company inc.</ResponsibleOrganization>
  <PreviewImage xlink:href="http://www.prs.net/data/3213-kLRt-9494.png" />
   <SymbolReferences>
    <Link xlink:href="http://www.prs.net/metadata/symbol/2324-7828-9342.xml" />
    <Link xlink:href="http://www.prs.net/metadata/symbol/9232-5353-9392.xml" />
  </SymbolReferences>
</StandardColor>
```

B.3.1.9.2 JS Example object

```
{
    identifier: "aaead-34347",
    name: "standardColor A554B",
    description: "An example standardColor object",
    responsibleOrganization: "Carmenta AB",
    previewImage:"http://www.images.com/previewImageColor.jpg",
    symbolReferences:["http://www.prs.net/metadata/symbols/2324-7828-9342.xml"]
}
```

B.3.2 Object Collections

All object collection types contain a set of objects of the type that the collection is of. For example, a SymbolSetCollection contains a set of SymbolSets, a RuleCollection contains a set of Rules. The collections are exposed to users through SearchResponses.

Note that the object collections do not exist in the objects returned as JSON, instead they are represented as arrays.

For an example of how the object collections are used in the interface, please refer to chapter B.3.5.3 (SearchResponse specification).

B.3.3 Reference Collections

For each of the object types described in chapter B.3.1, except for itemBase, a collection of Links exist. Each Link represents an URL pointing to a metadata object of the same type as the reference collection. For example, a SymbolSetReferences object contains Links pointing to SymbolSets.

Note that the reference collections do not exist in the objects returned as JSON, instead they are represented as arrays.

Reference collections are used to represent the relations between objects described in chapter 6.

B.3.4 Additional Objects

This chapter contains support objects, i.e. objects that are used in several other objects but not used on their own.

B.3.4.1 OnlineResource

The OnlineResource object contains a Link object, which is essentially a URL-container, and a list of content types which can be retrieved from the resource, via the link. If no content-types are listed, or no Link is provided, the OnlineResource is essentially not accessible.

This object will often have a different element name than “OnlineResource” in the returned metadata. See chapter B.3.4.1.1 for an example.

This object does not extend the itemBase object.

This object is most commonly used to represent the different ways of retrieving data and metadata as well as the available services in an interface.

Table 25: OnlineResource Fields

Field	Type	Description
Link	Link	A link, containing the URL for the OnlineResource.
ContentTypes	ContentTypes	The content types (mime types) that can be retrieved from the OnlineResource.

B.3.4.1.1 Xml Example object

In this example, the object is named “StandardColor”.

```
<StandardColor>
  <Link xlink:type="simple" xlink:href="http://www.restprs.net/data/standardColor/" />
  <ContentTypes>
    <ContentType>application/vnd.ogc.se+xml</ContentType>
  </ContentTypes>
</StandardColor>
```

B.3.4.1.2 JS Example object

```
{
    link:"http://www.restprs.net/metadata/123-123-123",
    contentTypes:["application/xml", "application/json"]
}
```

B.3.4.2 OnlineResources

Represents a collection of OnlineResources. This object does not exist in the JSON schema.

B.3.4.2.1 Xml Example Object

In this example, the OnlineResources object is named “Operations” and its OnlineResource objects are named after search operations.

```
<Operations>
  <GetStandardColor>
    <Link xlink:type="simple" />
    <ContentTypes />
  </GetStandardColor>
  <GetStandardFont>
    <Link xlink:type="simple"
xlink:href="http://www.restprs.net/services/search/standardFont.xml" />
    <ContentTypes>
      <ContentType>application/xml</ContentType>
      <ContentType>application/json</ContentType>
    </ContentTypes>
  </GetStandardFont>
</Operations>
```

B.3.4.3 Link

The Link object represents a URL, and does so by extending the simpleLink element of [Xlink]. In the JSON Schema, this object is not an Xlink extension, simply a string.

B.3.4.4 Endpoints

The Endpoints object is used in the capabilities document of the metadata and data resources.

The Endpoint object contains a specific set of OnlineResources:

- StandardColor
- StandardFont
- SymbolSet
- Symbol
- RuleSet
- Rule
- ApplicationSchema
- PreviewImages

Each of these objects contains a list of content types and a Link object. This object is used to represent endpoints both on the metadata and on the data.

B.3.4.4.1 XML Example object

```
<Endpoints>
  <StandardColor>
   <Link xlink:href="http://www.restprs.net/metadata/standardColor/" />
    <ContentTypes>
     <ContentType>application/xml</ContentType>
     <ContentType>application/json</ContentType>
   </ContentTypes>
  </StandardColor>
   <StandardFont>
   <Link xlink:href="http://www.restprs.net/metadata/standardFont/" />
    <ContentTypes>
     <ContentType>application/xml</ContentType>
     <ContentType>application/json</ContentType>
   </ContentTypes>
 </StandardFont>
  <SymbolSet>
   <Link />
   <ContentTypes />
 </SymbolSet>
  <Symbol>
   <Link xlink:href="http://www.restprs.net/metadata/Symbol/" />
    <ContentTypes>
     <ContentType>application/xml</ContentType>
     <ContentType>application/json</ContentType>
   </ContentTypes>
  </Symbol>
</Endpoints>
```

B.3.4.4.2 JS Example object

```
{
    standardColor:[],
       standardFont:[],
       symbolSet:[{link:"http://www.prs.net/metadata/symbolSets/",
contentTypes:["application/xml","application/json"]}],
       symbol:[],
       ruleSet:[{link:"http://www.prs.net/metadata/ruleSets/",
contentTypes:["application/xml","application/json"]}],
       rule:[],
       applicationSchema:[],
       previewImage:[]
}
```

B.3.4.5 Operations

The Operations object is used in the capabilities document of the search resource.

The Operations object contains one OnlineResource for each of the available searches, described in chapter B.4.6:

- GetStandardColor
- GetStandardFont
- GetSymbolSet
- GetSymbol
- GetRuleSet
- GetRule
- GetApplicationSchema

In the example below, only GetStandardFont has a Link and content types specified. To fully implement this specification one must specify at least one content type and provide a link for each operation.

B.3.4.5.1 XML Example object

```
   <Operations>
     <GetStandardColor>
      <Link />
      <ContentTypes />
    </GetStandardColor>
     <GetStandardFont>
      <Link xlink:href="http://www.restprs.net/services/search/standardFont.xml" />
       <ContentTypes>
        <ContentType>application/xml</ContentType>
        <ContentType>application/json</ContentType>
      </ContentTypes>
    </GetStandardFont>
     <GetSymbolSet>
      <Link />
      <ContentTypes />
    </GetSymbolSet>
     <GetSymbol>
      <Link />
      <ContentTypes />
    </GetSymbol>
     <GetRuleSet>
      <Link />
      <ContentTypes />
    </GetRuleSet>
     <GetRule>
      <Link />
      <ContentTypes />
    </GetRule>
     <GetApplicationSchema>
      <Link />
      <ContentTypes />
    </GetApplicationSchema>
  </Operations>
```

B.3.4.5.2 JS Example object

```
{
       "getStandardColor":[{link:"http://www.prs.net/search/standardColors.json",
contentTypes:["application/json"]}],
       "getStandardFont":[],
       "getSymbolSet":[{link:"http://www.prs.net/search/symbolSets.xml",
contentTypes:["application/xml"]}],
       "getSymbol":[],
       "getRuleSet":[{link:"http://www.prs.net/search/ruleSets.json",
contentTypes:["application/json"]}],
       "getRule":[],
       "getApplicationSchema":[]
}
```

B.3.5 Response Objects

This chapter specifies the objects that are used as return types in the REST interface.

B.3.5.1 ResponseBase

All other response objects extend the ResponseBase object. This object is not exposed to users in other ways than acting as the base for response objects.

Table 26: ResponseBase Fields

Field	Type	Description
Timestamp	dateTime	Date and time when response was sent

To view the XML schema for this object, refer to chapter B.6.

B.3.5.2 MetadataResponse

The MetadataResponse object represents a successful response from the metadata interface. This response is obtained when retrieving a single metadata item, all of which are described in chapter B.3.1.

Table 27: MetadataResponse Fields

Field	Type	Description
Item	Symbol | SymbolSet | Rule | RuleSet | ApplicationSchema | StandardFont | StandardColor	The metadata for the item with the matching identifier. At most one object is returned.
ResultCount	Int	The number of returned results.

To view the XML schema for this object, refer to chapter B.6.

B.3.5.2.1 XML Example object

```
<?xml version="1.0" ?>
 <MetadataResponse xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Timestamp>2012-10-24T11:04:19.6460908+02:00</Timestamp>
  <ResultCount>1</ResultCount>
   <Rule>
    <Identifier>923480230423098</Identifier>
    <Name>myRule</Name>
    <Description>Test rule</Description>
    <NativeEncoding>application/vnd.ogc.sld+xml</NativeEncoding>
     <FeatureTypes>
      <FeatureType>FeatureTypeZeta</FeatureType>
    </FeatureTypes>
     <SymbolReferences>
      <Link xlink:href="http://www.prs.net/metadata/symbol/3242542.xml" />
      <Link xlink:href="http://www.prs.net/metadata/symbol/3266254.xml" />
      <Link xlink:href="http://www.prs.net/metadata/symbol/3277754.xml" />
    </SymbolReferences>
     <ApplicationSchemaReferences>
      <Link xlink:href="http://www.prs.net/metadata/applicationSchema/324234.xml" />
      <Link xlink:href="http://www.prs.net/metadata/applicationSchema/399274.xml" />
      <Link xlink:href="http://www.prs.net/metadata/applicationSchema/377254.xml" />
    </ApplicationSchemaReferences>
    <RuleSetReferences />
  </Rule>
</MetadataResponse>
```

B.3.5.2.2 JS Example object

```
{
    timestamp:"2012-04-23:1532",
    resultItem: {
        identifier: "234234-364364",
        name: "anyName",
        description: "An example rule object",
        responsibleOrganization: "Carmenta AB",
        nativeEncoding:"application/vnd.google-earth.kml+xml",
           featureTypes:["BuiltUpArea"],
           symbolReferences:["http://www.restprs.net/symbols/34543-23423"],
           applicationSchemaReferences:["http://www.restprs.net/applicationSchemas/ee664-
23423"],
           ruleSetReferences:["http://www.restprs.net/ruleSets/aaeefa-23423"]
    },
    resultCount:1
}
```

B.3.5.3 SearchResponse

The SearchResponse object is returned when a search has been successfully performed. It contains a collection whose type is dependent on the type of search performed. For more information about available searches, see chapter B.4.6.

Table 28: SearchResponse Fields

Field	Type	Description
Item	SymbolCollection | SymbolSetCollection | RuleCollection | RuleSetCollection | ApplicationSchemaCollection | StandardFontCollection | StandardColorCollection	A collection of metadata items matching the performed search.
ResultCount	Int	The number of returned results

To view the XML schema for this object, refer to chapter B.6.

B.3.5.3.1 XML Example object

```
<?xml version="1.0" ?>
 <SearchResponse xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Timestamp>2012-10-24T11:04:19.6460908+02:00</Timestamp>
  <ResultCount>2</ResultCount>
   <RuleCollection>
    <Rules>
      <Rule>
       <Identifier>923480230423098</Identifier>
       <Name>myRule</Name>
       <Description>Test rule</Description>
       <NativeEncoding>application/vnd.ogc.sld+xml</NativeEncoding>
        <FeatureTypes>
         <FeatureType>FeatureTypeZeta</FeatureType>
       </FeatureTypes>
        <SymbolReferences>
         <Link xlink:href="http://www.prs.net/metadata/symbol/3242542.xml" />
         <Link xlink:href="http://www.prs.net/metadata/symbol/3266254.xml" />
         <Link xlink:href="http://www.prs.net/metadata/symbol/3277754.xml" />
       </SymbolReferences>
        <ApplicationSchemaReferences>
         <Link xlink:href="http://www.prs.net/metadata/applicationSchema/324234.xml" />
         <Link xlink:href="http://www.prs.net/metadata/applicationSchema/399274.xml" />
         <Link xlink:href="http://www.prs.net/metadata/applicationSchema/377254.xml" />
       </ApplicationSchemaReferences>
       <RuleSetReferences />
      </Rule>
    </Rules>
  </RuleCollection>
</SearchResponse>
```

B.3.5.3.2 JS Example object

```
{
    timestamp:"2012-04-05:1733",
    resultItems:[{
        identifier: "234234-364364",
        name: "anyName",
        description: "An example rule object",
        responsibleOrganization: "Carmenta AB",
        nativeEncoding:"application/vnd.google-earth.kml+xml",
           featureTypes:["BuiltUpArea"],
           symbolReferences:["http://www.restprs.net/symbols/34543-23423"],
           applicationSchemaReferences:["http://www.restprs.net/applicationSchemas/ee664-
23423"],
           ruleSetReferences:["http://www.restprs.net/ruleSets/aaeefa-23423"]
    }],
    resultCount:1
}
```

B.3.5.4 ErrorReport

An ErrorReport is obtained whenever a request resulted in some kind of error. The error may have been caused by incorrect input from the client or an internal server error.

Table 29: ErrorReport Fields

Field	Type	Description
ErrorTitle	string	A descriptive title.
ErrorDescription	string	A detailed description of the error and measures that can be taken, if any exist, to avoid the error.

To view the XML schema for this object, refer to chapter B.6.

B.3.5.4.1 XML Example object

```
<?xml version="1.0"?>
<ErrorReport xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Timestamp>2012-04-24T09:48:59.9940728+02:00</Timestamp>
  <ErrorTitle>Internal server error</ErrorTitle>
  <ErrorDescription>An internal server error ocurred during metadata
retrieval.</ErrorDescription>
</ErrorReport>
```

B.3.5.4.2 JS Example object

```
{
    timestamp:"2012-05-30:1655",
    errorTitle:"Invalid identifier",
    errorDescription:"No object with the specified identifier found."
}
```

B.3.6 Capabilities documents

This section specifies the structure of the documents that are returned from the capabilities resource of the respective root resources.

B.3.6.1 MetadataCapabilities

The MetadataCapabilities objects purpose is to list the capabilities of the metadata interface in a concise manner. The MetadataCapabilities object extends the ResponseBase object and adds the following fields.

Table 30: MetadataCapabilities Fields

Field	Type	Description
Title	string	A descriptive title.
Description	string	A human readable description of the returned capabilities.
Endpoints	Endpoints	A collection of metadata endpoints. (see B.3.4.4)

Conformant implementations shall provide a Link for all endpoints specified in the MetadataCapabilities document. The example objects below omit some Links for brevity.

To view the XML schema for this object, refer to chapter B.6.

B.3.6.1.1 Xml Example object

```
<?xml version="1.0" ?>
 <MetadataCapabilities xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Timestamp>2012-10-24T11:04:19.6460908+02:00</Timestamp>
  <Title>The metadata interface</Title>
  <Description>This is the interface through which metadata is retrieved.</Description>
   <Endpoints>
     <StandardColor>
      <Link xlink:href="http://www.restprs.net/metadata/standardColor/" />
       <ContentTypes>
        <ContentType>application/xml</ContentType>
        <ContentType>application/json</ContentType>
      </ContentTypes>
    </StandardColor>
     <StandardFont>
      <Link xlink:href="http://www.restprs.net/metadata/standardFont/" />
       <ContentTypes>
        <ContentType>application/xml</ContentType>
        <ContentType>application/json</ContentType>
      </ContentTypes>
    </StandardFont>
     <Symbol>
      <Link xlink:href="http://www.restprs.net/metadata/Symbol/" />
       <ContentTypes>
        <ContentType>application/xml</ContentType>
        <ContentType>application/json</ContentType>
      </ContentTypes>
    </Symbol>
  </Endpoints>
</MetadataCapabilities>
```

B.3.6.1.2 JS Example object

```
{
    timeStamp: "2012-04-23:1555",
    name: "Metadata Capabilities",
    description:"An interface from which to retrieve metadata",
    endpoints: {
        standardColor:[],
           standardFont:[],
           symbolSet:[{link:"http://www.prs.net/metadata/symbolSets/",
contentTypes:["application/xml","application/json"]}],
           symbol:[],
           ruleSet:[{link:"http://www.prs.net/metadata/ruleSets/",
contentTypes:["application/xml","application/json"]}],
           rule:[],
           applicationSchema:[],
           previewImage:[]
    }
}
```

B.3.6.2 DataCapabilities

The DataCapabilities objects purpose is to list the capabilities of the data interface in a concise manner. The DataCapabilities object extends the ResponseBase object and adds the following fields.

Table 31: DataCapabilities Fields

Field	Type	Description
Title	string	A descriptive title.
Description	string	A human readable description of the returned capabilities.
Endpoints	Endpoints	A collection of data endpoints. (see B.3.4.4)

To view the XML schema for this object, refer to chapter B.6.

Note that conformant implementations are not required to provide a Link for all endpoints specified in the data capabilities – it must not be possible to retrieve data for all object types.

B.3.6.2.1 XML Example object

```
<?xml version="1.0" ?>
 <DataCapabilities xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Timestamp>2012-10-24T11:04:19.6460908+02:00</Timestamp>
  <Title>Data capabilities</Title>
  <Description>The interface through which data is retrieved</Description>
   <Endpoints>
     <StandardColor>
      <Link xlink:href="http://www.restprs.net/data/" />
       <ContentTypes>
        <ContentType>application/vnd.ogc.se+xml</ContentType>
      </ContentTypes>
    </StandardColor>
     <StandardFont>
      <Link />
      <ContentTypes />
    </StandardFont>
     <SymbolSet>
      <Link />
      <ContentTypes />
    </SymbolSet>
     <Symbol>
      <Link />
      <ContentTypes />
    </Symbol>
     <RuleSet>
      <Link />
      <ContentTypes />
    </RuleSet>
     <Rule>
      <Link />
      <ContentTypes />
    </Rule>
     <ApplicationSchema>
      <Link />
      <ContentTypes />
    </ApplicationSchema>
     <PreviewImages>
      <Link xlink:href="http://www.restprs.net/data/" />
       <ContentTypes>
        <ContentType>image/png</ContentType>
        <ContentType>image/gif</ContentType>
        <ContentType>image/jpeg</ContentType>
      </ContentTypes>
    </PreviewImages>
  </Endpoints>
</DataCapabilities>
```

B.3.6.2.2 JS Example object

```
{
    timestamp:"2012-04-05:1433",
    name:"Data Capabilities",
    description:"Interface for retrieving data",
    endpoints: {
        standardColor:[],
           standardFont:[],
           symbolSet:[{link:"http://www.prs.net/data/", contentTypes:["vnd.google-
earth.kml+xml"]}],
           symbol:[],
           ruleSet:[{link:"http://www.prs.net/data/", contentTypes:["vnd.google-
earth.kml+xml"]}],
           rule:[],
           applicationSchema:[],
           previewImage:[]
    }
}
```

B.3.6.3 SearchCapabilities

The SearchCapabilities object describes the capabilities of the search interface.

SearchCapabilities extends the ResponseBase object and adds the following fields.

Table 32: SearchCapabilities Fields

Field	Type	Description
Title	string	A descriptive title.
Description	string	A human readable description of the returned capabilities.
Link	Link	A Link to the service itself.
Operations	Operations	A collection of OnlineResources, each representing one Operation.
SubServices	OnlineResources	A collection of resources linking to subservices.

Conformant implementations shall provide a Link for all Operations specified in the SearchCapabilities document. The example objects below omit some Links for brevity.

B.3.6.3.1 XML Example Object

```
<?xml version="1.0" ?>
 <SearchCapabilities xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://dgiwg.org/schemas/prs/rest">
  <Timestamp>2012-10-24T11:04:19.6460908+02:00</Timestamp>
  <Title>Search interface</Title>
  <Description>The search interface. This interface hosts metadata searches.</Description>
  <Link xlink:href="http://www.restprs.net/search/" />
   <Operations>
     <GetStandardColor>
      <Link />
      <ContentTypes />
    </GetStandardColor>
     <GetStandardFont>
      <Link xlink:href="http://www.restprs.net/search/standardFonts" />
       <ContentTypes>
        <ContentType>application/xml</ContentType>
        <ContentType>application/json</ContentType>
      </ContentTypes>
    </GetStandardFont>
     <GetSymbolSet>
      <Link />
      <ContentTypes />
    </GetSymbolSet>
     <GetSymbol>
      <Link />
      <ContentTypes />
    </GetSymbol>
     <GetRuleSet>
      <Link />
      <ContentTypes />
    </GetRuleSet>
     <GetRule>
      <Link />
      <ContentTypes />
    </GetRule>
     <GetApplicationSchema>
      <Link />
      <ContentTypes />
    </GetApplicationSchema>
  </Operations>
</SearchCapabilities>
```

B.3.6.3.2 JS Example object

```
{
    timestamp:"20010203T0405+0100",
    title:"Restful PRS search interface",
    description:"A restful prs, serving metadata searches",
    link:"http://www.restfulprs.net/search/",
    operations: {
           "getStandardColor":[{link:"http://www.prs.net/search/standardColors",
contentTypes:["application/json", “application/xml”]}],
           "getStandardFont":[],
           "getSymbolSet":[{link:"http://www.prs.net/search/symbolSets",
contentTypes:["application/xml"]}],
           "getSymbol":[],
           "getRuleSet":[{link:"http://www.prs.net/search/ruleSets",
contentTypes:["application/json"]}],
           "getRule":[],
           "getApplicationSchema":[]
    },
    subServices: []
}
```

B.4 Interface specification

This chapter specifies the interface – the means for retrieving metadata, searching for metadata and retrieving data.

B.4.1 Notation used in the interface specification

URL Patterns are specified according to the IETF URI Template, as specified in [RFC6570]. Variable values are specified using regular expressions.

Throughout this chapter the {restfulprsdomain} element represents the domain of the restful web service. Example domains are http://www.restfulprs.net or http://company.publicservices.restful.net . The {restfulprsdomain} can also contain subdirectories, e.g. http://www.restfulprs.net/restinterface/.

The cardinalities used in this section can be translated using the following list:

- * - Zero or more.
- 1 – Exactly one.
- 0..1 – Zero or one.

A brief description of the purpose and contents of each table in the following subchapters is provided in the table below.

Table 33: Interface specification table description

Table	Description
URL	The URL table contains the URL pattern of an endpoint and example URLs.
Request variables	Describes the variables available for a certain request, the cardinality of the variables and potential limitations in variable content.
Request headers	Lists all HTTP headers for the request that has some specific purpose in this RESTful implementation and their allowed values.
Methods	Lists the available methods and what action a certain method represents. If a method is not listed here its use and effect will be implementation specific.
Response codes	Lists methods, response codes available for that method and what type of object is returned depending on response code.
Response headers	Lists all HTTP headers for the response that has some specific purpose in this RESTful implementation.

B.4.2 Mandatory metadata output formats

The metadata and search interfaces described in the upcoming chapters support a ‘format’ parameter which provides a simplified way to specify the format of the returned metadata resouces.

The table below lists the valid values for the format parameter, the corresponding MIME types and the expected format of the returned resources.

All implementations shall support these three formats. Implementations may choose to support additional formats – these can be advertised through the MetadataCapabilities and SearchCapabilities documents.

Table 34: Mandatory metadata formats

Format parameter	MIME type	Schema	Description
.xml	application/xml	B.6	An xml representation of the returned metadata resource according to the XML schema in this annex.
.json	application/json	B.7	A JSON representation of the returned metadata resource according to the JSON schema in this annex.
.html	text/html	N/A	A human-readable representation of the returned metadata resource, formatted as HTML5.

B.4.3 Default metadata output format

For the metadata and search interfaces:

If neither a format parameter nor a content-type is specified in the request, the metadata resource shall be returned as a human-readable representation, using the text/html content type.

For the data interface:

If a content-type parameter is not specified, the data shall be returned in its native encoding.

B.4.4 Data and Metadata in the REST interface

The metadata object representations that are returned contain no reference to the data of the corresponding object. That is, metadata objects are not directly linked to data objects.

The two are however indirectly linked by a governing rule that exists in the REST interface:

The identifier property of a metadata object is used to retrieve the corresponding data item.

Example:

- http://www.restprs.net/metadata/symbols/4566-3442.xml
  - retrieves metadata for the symbol with the identifier property “4566-3442” in XML format.

- http://www.restprs.net/data/4566-3442?content-type=application/vnd.google-earth.kml+xml
  - retrieves the data for the symbol with the identifier property “4566-3442” in KML format.

In conclusion: retrieving the data for some piece of metadata is performed by creating an url according to this pattern: {baseUrl}{identifier}{?content-type} where the identifier is the same as the identifier property of the metadata item for which you want to retrieve the data.

B.4.5 Metadata interface

This chapter describes the ‘metadata’ root resource, which provides a way to access individual metadata items.

In contrast to performing searches, described in chapter B.4.6, the metadata retrieval methods described in this chapter only have one parameter, the identifier of an object. The reason for this distinction is simple: to provide a RESTful method of retrieving individual metadata objects whilst also providing a comprehensive way to perform searches.

B.4.5.1 StandardColor

Corresponds to the operation described in 7.1.1 - GetStandardColors. This is a more restricted version where the only available parameters are the identifier of the StandardColor and the return type.

URL	
Pattern	{restfulprsdomain}/metadata/standardColors/{identifier}{.format}{?content-type}
Examples	http://www.prs.net/metadata/standardColors/6346.xml http://www.restfulprs.net/metadata/standardColors/hJE5-JF89.json http://www.restfulprs.net/metadata/standardColors/hJE5-JF89.xml?content-type = text/html

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	A unique identifier for a StandardColor.
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
HEAD	Check whether the StandardColor exists.
GET	Retrieve the metadata for the StandardColor.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	A StandardColor with the specified ID exists.	Headers, no message-body.
	404 Not Found	No StandardColor with the specified ID exists.	Headers, no message-body.

	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, no message-body.
GET	200 OK	A StandardColor was found and returned.	Headers, MetadataResponse containing the matching StandardColor, see chapter 0.
	404 Not Found	No StandardColor with the specified ID exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.5.2 StandardFont

Corresponds to the operation described in 7.1.2 - GetStandardFonts. This is a more restricted version where the only available parameters are the identifier of the StandardFont and its format.

URL	
Pattern	{restfulprsdomain}/metadata/standardFonts/{identifier}{.format}{?content-type}
Examples	http://www.prs.net/metadata/standardFonts/63463.json http://www.prs.net/metadata/standardFonts/hJE5-J339.xml http://www.prs.net/metadata/standardFonts/hJE5-J339?content-type=text/html

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	A unique identifier for a StandardFont.
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
HEAD	Check whether the StandardFont exists.
GET	Retrieve the metadata for the StandardFont.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	A StandardFont with the specified ID exists.	Headers, no message-body.
	404 Not Found	No StandardFont with the specified ID exists.	Headers, no message-body.
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, no message-body.
GET	200 OK	A StandardFont was found and returned.	Headers, MetadataResponse containing the matching StandardFont, see chapter 0.

	404 Not Found	No StandardFont with the specified ID exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.5.3 SymbolSet

Corresponds to the operation described in 7.1.3 - GetSymbolSets. This is a more restricted version where the only available parameters are the identifier of the SymbolSet and its format.

URL	
Pattern	{restfulprsdomain}/metadata/symbolSets/{identifier}{.format}{?content-type}
Examples	http://www.prs.net/metadata/symbolSets/63443.json http://www.prs.net/metadata/symbolSets/hJE5-JAA9.xml

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	A unique identifier for a SymbolSet.
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
HEAD	Check whether the SymbolSet exists.
GET	Retrieve the metadata for the SymbolSet.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	A SymbolSet with the specified ID exists.	Headers, no message-body.
	404 Not Found	No SymbolSet with the specified ID exists.	Headers, no message-body.
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, no message-body.
GET	200 OK	A SymbolSet was found and returned.	Headers, MetadataResponse containing the matching SymbolSet, see chapter 0.
	404 Not Found	No SymbolSet with the specified ID exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0

PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.5.4 Symbol

Corresponds to the operation described in 7.1.4 - GetSymbols. This is a more restricted version where the only available parameters are the identifier of the Symbol and its format.

URL	
Pattern	{restfulprsdomain}/metadata/symbols/{identifier}{.format}{?content-type}
Examples	http://www.prs.net/metadata/symbols/63663.xml http://www.prs.net/metadata/symbols/hJE5-F339.json

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	A unique identifier for a Symbol.
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
HEAD	Check whether the Symbol exists.
GET	Retrieve the metadata for the Symbol.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	A Symbol with the specified ID exists.	Headers, no message-body.
	404 Not Found	No Symbol with the specified ID exists.	Headers, no message-body.
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, no message-body.

GET	200 OK	A Symbol was found and returned.	Headers, MetadataResponse containing the matching Symbol, see chapter 0.
	404 Not Found	No Symbol with the specified ID exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.5.5 RuleSet

Corresponds to the operation described in 7.1.5 - GetRuleSets. This is a more restricted version where the only available parameters are the identifier of the RuleSet and its format.

URL	
Pattern	{restfulprsdomain}/metadata/ruleSets/{identifier}{.format}{?content-type}
Examples	http://www.prs.net/metadata/ruleSets/77343.xml http://www.prs.net/metadata/ruleSets/hJE5-HR55.json

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	A unique identifier for a RuleSet.
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
HEAD	Check whether the RuleSet exists.
GET	Retrieve the metadata for the RuleSet.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	A RuleSet with the specified ID exists.	Headers, no message-body.
	404 Not Found	No RuleSet with the specified ID exists.	Headers, no message-body.
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, no message-body.
GET	200 OK	A RuleSet was found and returned.	Headers, MetadataResponse containing the matching RuleSet, see chapter 0.
	404 Not Found	No RuleSet with the specified ID exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.5.6 Rule

Corresponds to the operation described in 7.1.6 - GetRules. This is a more restricted version where the only available parameters are the identifier of the Rule and its format.

URL	
Pattern	{restfulprsdomain}/metadata/rules/{identifier}{.format}{?content-type}
Examples	http://www.prs.net/metadata/rules/32243.xml http://www.prs.net/metadata/rules/hJE5-88YT.json

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	A unique identifier for a Rule.
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.

content-type		cardinality	value	description
		0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
HEAD	Check whether the Rule exists.
GET	Retrieve the metadata for the Rule.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	A Rule with the specified ID exists.	Headers, no message-body.
	404 Not Found	No Rule with the specified ID exists.	Headers, no message-body.
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, no message-body.
GET	200 OK	A Rule was found and returned.	Headers, MetadataResponse containing the matching Rule, see chapter 0.
	404 Not Found	No Rule with the specified ID exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.5.7 ApplicationSchema

Corresponds to the operation described in 7.1.7 - GetApplicationSchemas. This is a more restricted version where the only available parameters are the identifier of the ApplicationSchema and its format.

URL	
Pattern	{restfulprsdomain}/metadata/applicationSchemas/{identifier}{.format}{?content-type}
Examples	http://www.prs.net/metadata/applicationSchemas/63443.xml http://www.prs.net/metadata/applicationSchemas/hJE5-JAA9.json

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	A unique identifier for an ApplicationSchema.
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
HEAD	Check whether the ApplicationSchema exists.
GET	Retrieve the metadata for the ApplicationSchema.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	An ApplicationSchema with the specified ID exists.	Headers, no message-body.
	404 Not Found	No ApplicationSchema with the specified ID exists.	Headers, no message-body.
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, no message-body.
GET	200 OK	An ApplicationSchema was found and returned.	Headers, MetadataResponse containing the matching ApplicationSchema, see chapter 0.

	404 Not Found	No ApplicationSchema with the specified ID exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.5.8 Metadata Capabilities

The capabilities operation returns an object describing the capabilities of the metadata.

URL	
Pattern	{restfulprsdomain}/metadata/capabilities.{format}
Examples	http://www.prs.net/metadata/capabilities.xml http://www.prs.net/metadata/capabilities.json

Request variables			
format	**cardinality**	**value**	**description**
	1	json | xml | html	The format in which to return the capabilities result – see B.4.2.

Methods	
Method	**Description**
HEAD	Not allowed.
GET	Retrieve the capabilities of the metadata.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0
GET	200 OK	Capabilities returned.	Headers, A MetadataCapabilities object, see chapter 0.
	406 Not Acceptable	The specified format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6 Search interface

This chapter describes the ‘search’ root resource, which provides a way to perform searches for metadata items.

This chapter lists all mandatory search operations. The value of a parameter is specified using regular expressions. Parameters are appended to requests using the query string.

The cardinalities used in this section can be translated using the following list:

- * - Several or none.
- 1 – Exactly one.
- 0..1 – Zero or one.

II a parameter has a cardinality of “*” it must be represented, in the query string, as a csv-value. For example: the name parameter of the StandardColors search could be represented as “?name=tac,tactical,nautical”. When such a csv-list is provided, the search shall return any result matching at least one of the items in the list.

In contrast to the methods described in B.4.5 , the methods described in this chapter have considerably more input options but can also return several results. All searches still return metadata, however. The purpose of this separation is to provide a RESTful method of retrieving metadata whilst also providing a comprehensive way to perform searches and still conforming to the REST architecture.

If a search method does not have any mandatory parameters, specifying none of the optional parameters will return as many objects matching the search as the implementation permits. It is recommended to enforce a maximum number of results even if the client does not specify the maxResults parameter. It is also recommended to enforce a global maxResults number, overriding the maxResults parameter, to avoid performance issues.

B.4.6.1 StandardColors

Corresponds to the operation described in 7.1.1 - GetStandardColors. This is a less restricted version, lacking the identifier parameter but complemented by other parameters.

URL	
Pattern	{restfulprsdomain}/search/standardColors{.format}{?name,keyword,maxResults,encoding,content-type}
Examples	http://www.prs.net/search/standardColors.xml?name=sea,tac http://www.prs.net/search/standardColors.json http://www.prs.net/search/standardColors.json?keyword=sea&maxResults=50

Request variables			
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
name	**cardinality**	**value**	**description**
	*	.+	A non-unique name, or a substring of a name, of a StandardColor

keyword	**cardinality**	**value**	**description**
	*	.+	A keyword, used to search through all human readable fields of the StandardColor.
maxResults	**cardinality**	**value**	**description**
	0..1	[0-9]+	The maximum number of results to be retrieved. If this variable is not specified, some default setting shall be used.
startAt	**cardinality**	**value**	**description**
	0..1	[0-9]+	The result number to start retrieval from. If this number is greater than the number of matching results, an empty ApplicationSchemaCollection shall be returned.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
GET	Retrieve a collection of StandardColors matching the provided parameters
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
GET	200 OK	A search was successfully performed, all matching results returned.	Headers, SearchResponse containing a StandardColorCollection, see chapter B.3.5.3
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD,PUT,POST,DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.

HEAD,PUT, POST,DELETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6.2 StandardFonts

Corresponds to the operation described in 7.1.2 - GetStandardFonts. This is a less restricted version, lacking the identifier parameter but complemented by other parameters.

URL	
Pattern	{restfulprsdomain}/search/standardFonts{.format}{?name,keyword,maxResults,content-type}
Examples	http://www.prs.net/search/standardFonts.xml?name=tactical&maxResults=100 http://www.prs.net/search/standardFonts.json?keyword=tactical&content-type=text/html

Request variables			
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
name	**cardinality**	**value**	**description**
	*	.+	A non-unique name, or a substring of a name, of a StandardFont
keyword	**cardinality**	**value**	**description**
	*	.+	A keyword, used to search through all human readable fields of the StandardFont.
maxResults	**cardinality**	**value**	**description**
	0..1	[0-9]+	The maximum number of results to be retrieved. If this variable is not specified, some default setting shall be used.
startAt	**cardinality**	**value**	**description**
	0..1	[0-9]+	The result number to start retrieval from. If this number is greater than the number of matching results, an empty ApplicationSchemaCollection shall be returned.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
GET	Retrieve a StandardFontsCollection matching the provided parameters

HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
GET	200 OK	A search was successfully performed, all matching results returned.	Headers, SearchResponse containing a standardFontCollection, see chapter B.3.5.3
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DELETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6.3 SymbolSets

Corresponds to the operation described in 7.1.3 - GetSymbolSets. This is a less restricted version, lacking the identifier parameter but complemented by other parameters.

URL	
Pattern	{restfulprsdomain}/search/symbolSets{.format}{?name,keyword,maxResults,responsibleOrganisationName,content-type}
Examples	http://www.prs.net/search/symbolSets.xml?name=tactical http://www.prs.net/search/symbolSets?keyword=tactical&content-type=text/html

Request variables			
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
name	**cardinality**	**value**	**description**
	*	.+	A non-unique name, or a substring of a name, of a SymbolSet
keyword	**cardinality**	**value**	**description**
	*	.+	A keyword, used to search through all human readable fields of the SymbolSet.
maxResults	**cardinality**	**value**	**description**
	0..1	[0-9]+	The maximum number of results to be retrieved. If this variable is not specified, some default setting shall be used.
startAt	**cardinality**	**value**	**description**
	0..1	[0-9]+	The result number to start retrieval from. If this number is greater than the number of matching results, an empty ApplicationSchemaCollection shall be returned.
responsibleOrganisationName	**cardinality**	**value**	**description**
	*	.+	A name, or substring of a name, of an organisation. All SymbolSets with a matching, or partially matching, responsibleOrganisationName will be returned.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**

GET	Retrieve a SymbolSetCollection matching the provided parameters
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
GET	200 OK	A search was successfully performed, all matching results returned.	Headers, SearchResponse containing a SymbolSetCollection, see chapter B.3.5.3
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD,PUT, POST,DEL ETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DEL ETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6.4 Symbols

Corresponds to the operation described in 7.1.4 - GetSymbols. This is a less restricted version, lacking the identifier parameter but complemented by other parameters.

URL	
Pattern	{restfulprsdomain}/search/symbols{.format}{?name,keyword,maxResults,symbolSetId,symbolType,content-type}
Examples	http://www.prs.net/search/symbols.xml?name=tactical http://www.prs.net/search/symbols.json?maxResults=77

Request variables			
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
name	**cardinality**	**value**	**description**
	*	.+	A non-unique name, or a substring of a name, of a Symbol
keyword	**cardinality**	**value**	**description**
	*	.+	A keyword, used to search through all human readable fields of the Symbol.
maxResults	**cardinality**	**value**	**description**
	0..1	[0-9]+	The maximum number of results to be retrieved. If this variable is not specified, some default setting shall be used.
startAt	**cardinality**	**value**	**description**
	0..1	[0-9]+	The result number to start retrieval from. If this number is greater than the number of matching results, an empty ApplicationSchemaCollection shall be returned.
symbolSetId	**cardinality**	**value**	**description**
	*	[a-zA-Z0-9_-]+	The unique identifier of a SymbolSet. All returned symbols shall be part of that SymbolSet, if it exists. If the specified SymbolSet does not exist, no Symbols shall be returned – instead an error message of type ErrorReport <REF> shall be returned.
symbolType	**cardinality**	**value**	**description**
	*	[a-zA-Z0-9_-]+	A symbolType, or a substring of a symbolType. All Symbols with a matching, or partially matching, symbolType will be returned. See chapter <REF> for more information on symbolTypes.

content-type		cardinality	value	description
		0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
GET	Retrieve a SymbolCollection matching the provided parameters
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
GET	200 OK	A search was successfully performed, all matching results returned.	Headers, SearchResponse containing a SymbolCollection, see chapter B.3.5.3
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD,PUT, POST,DEL ETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DEL ETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6.5 RuleSets

Corresponds to the operation described in 7.1.5 - GetRuleSets. This is a less restricted version, lacking the identifier parameter but complemented by other parameters.

URL	
Pattern	{restfulprsdomain}/search/ruleSets{.format}{?name,keyword,maxResults,applicationSchemaId,responsibleOrganisationName,content-type }
Examples	http://www.prs.net/search/ruleSets.xml?responsibleOrganisationName=OGC http://www.prs.net/search/ruleSets.json?maxResults=77

Request variables

format	cardinality	value	description
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.

name	cardinality	value	description
	*	.+	A non-unique name, or a substring of a name, of a RuleSet

keyword	cardinality	value	description
	*	.+	A keyword, used to search through all human readable fields of the RuleSet.

maxResults	cardinality	value	description
	0..1	[0-9]+	The maximum number of results to be retrieved. If this variable is not specified, some default setting shall be used.

startAt	cardinality	value	description
	0..1	[0-9]+	The result number to start retrieval from. If this number is greater than the number of matching results, an empty ApplicationSchemaCollection shall be returned.

applicationSchemaId	cardinality	value	description
	*	[a-zA-Z0-9_-]+	The unique identifier of an ApplicationSchema. All returned RuleSets shall be compliant with that ApplicationSchema, if it exists. If the specified ApplicationSchema does not exist, no RuleSets shall be returned – instead an error message of type ErrorReport <REF> shall be returned.

responsibleOrganisationName	cardinality	value	description
	*	.+	A name, or substring of a name, of an organisation. All RuleSets with a matching, or partially matching, responsibleOrganisationName will be returned.

content-type		cardinality	value	description	
		0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.	

Methods	
Method	**Description**
GET	Retrieve a RuleSetCollection matching the provided parameters
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
GET	200 OK	A search was successfully performed, all matching results returned.	Headers, SearchResponse containing a RuleSetCollection, see chapter B.3.5.3
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD, PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DEL ETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6.6 Rules

Corresponds to the operation described in 7.1.6 - GetRules. This is a less restricted version, lacking the identifier parameter but complemented by other parameters.

URL	
Pattern	{restfulprsdomain}/search/rules{.format}{?name,keyword,maxResults,ruleSetId,featureType,content-type }
Examples	http://www.prs.net/search/rules.json?name=air http://www.prs.net/search/rules.xml?maxResults=77&name=sea&startAt=21

Request variables			
Format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
Name	**cardinality**	**value**	**description**
	*	.+	A non-unique name, or a substring of a name, of a Rule
keyword	**cardinality**	**value**	**description**
	*	.+	A keyword, used to search through all human readable fields of the Rule.
maxResults	**cardinality**	**value**	**description**
	0..1	[0-9]+	The maximum number of results to be retrieved. If this variable is not specified, some default setting shall be used.
startAt	**cardinality**	**value**	**description**
	0..1	[0-9]+	The result number to start retrieval from. If this number is greater than the number of matching results, an empty ApplicationSchemaCollection shall be returned.
ruleSetId	**cardinality**	**value**	**description**
	*	[a-zA-Z0-9_-]+	The unique identifier of a RuleSet. All returned Rules shall be part of that RuleSet, if it exists. If the specified RuleSet does not exist, no Rules shall be returned – instead an error message of type ErrorReport <REF> shall be returned.
featureType	**cardinality**	**value**	**description**
	*	.+	A featureType, or a substring of a featureType. Only Rules with a matching, or partially matching, featureType will be returned.

content-type		cardinality	value	description
		0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
GET	Retrieve a RuleCollection <REF> matching the provided parameters
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
GET	200 OK	A search was successfully performed, all matching results returned.	Headers, SearchResponse containing a RuleCollection, see chapter B.3.5.3
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD,PUT, POST,DEL ETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DEL ETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6.7 ApplicationSchemas

Corresponds to the operation described in 7.1.7 - GetApplicationSchemas. This is a less restricted version, lacking the identifier parameter but complemented by other parameters.

URL	
Pattern	{restfulprsdomain}/search/applicationSchemas{.format}{?name,keyword,maxResults,content-type }
Examples	http://www.prs.net/search/applicationSchemas.xml?name=air http://www.prs.net/search/applicationSchemas.json?maxResults=77&name=chart

Request variables			
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.
name	**cardinality**	**value**	**description**
	*	.+	A non-unique name, or a substring of a name, of a Rule
keyword	**cardinality**	**value**	**description**
	*	.+	A keyword, used to search through all human readable fields of the Rule.
maxResults	**cardinality**	**value**	**description**
	0..1	[0-9]+	The maximum number of results to be retrieved. If this variable is not specified, some default setting shall be used.
startAt	**cardinality**	**value**	**description**
	0..1	[0-9]+	The result number to start retrieval from. If this number is greater than the number of matching results, an empty ApplicationSchemaCollection shall be returned.
content-type	**cardinality**	**value**	**description**
	0..1	<Implementation dependent, must be some mime type that the metadata can deliver.>	The mime type in which to return the metadata. If used with the format parameter, this parameter has precedence.

Methods	
Method	**Description**
GET	Retrieve an ApplicationSchemaCollection, see chapter B.3.5.3, matching the provided parameters
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**

GET	200 OK	A search was successfully performed, all matching results returned.	Headers, SearchResponse containing a ApplicationSchemaCollection, see chapter B.3.5.3
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD,PUT, POST,DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DELETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.6.8 Search Capabilities

This operation describes the capabilities of the search interface. It does so through the use if a SearchCapabilities object, described in chapter 0.

URL	
Pattern	{restfulprsdomain}/search/capabilities.{format}
Examples	http://www.prs.net/search/capabilities.xml http://www.prs.net/search/capabilities.json

Request variables			
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata – see B.4.2.

Methods	
Method	**Description**
GET	Retrieve capabilities.
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**

GET	200 OK	Retrieve service capabilities	Headers, a SearchCapabilities, see chapter 0, describing the capabilities of the search interface.
	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD,PUT, POST,DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DELETE	405 Not Allowed	Allow	The allowed methods: GET

B.4.7 Data interface

There is only one way to retrieve data, and this specification only requires implementation of retrieval of one data item at a time.

It is up to each implementation of this interface what objects to allow data retrieval for and what MIME types to support – this is indicated in the data capabilities document.

If data retrieval of an unsupported type is attempted, the response shall be a 406 Not Acceptable with an ErrorReport.

B.4.7.1 Data retrieval

Corresponds to the operation described in 7.2.1 - GetData.

URL	
Pattern	{restfulprsdomain}/data/{identifier}{?content-type}
Examples	http://www.prs.net/data/2354-2678-5567 http://www.prs.net/data/23549845

Request variables			
identifier	**cardinality**	**value**	**description**
	1	[a-zA-Z0-9_-]+	The unique identifier of the data item to retrieve.
content-type	**cardinality**	**value**	**description**
	0..1	<Data dependent, must be some mime type that the data can deliver.>	The mime type in which you want the resource to be returned. If the specified format is not supported, an ErrorReport will be returned. If this parameter is not specified the data will be returned in its nativeEncoding.

Methods	
Method	**Description**
HEAD	Check whether an object with the specified identifier exists and if data retrieval is allowed for the object type.
GET	Retrieve the data.
PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
HEAD	200 OK	Data with the specified identifier, and in the specified format, is available.	Headers, no message-body.
	404 Not Found	No data with the specified identifier exists.	Headers, no message-body.
	406 Not Acceptable	The requested format cannot be delivered.	Headers, no message-body.

GET	200 OK	Data with the specified identifier was found and returned	Headers, message body containing the data.
	404 Not Found	No data with the specified identifier exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format cannot be delivered.	Headers, An ErrorReport, see chapter 0
PUT,POST, DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-Type	The format of the returned data. If used, this will most likely be some type of vendor specific content-encoding, such as “application/vnd.google-earth.kml+xml”.
PUT,POST, DELETE	405 Not Allowed	Allow	The allowed methods: HEAD, GET

B.4.7.2 Data Capabilities

This operation describes the capabilities of the data. It does so through the use of a DataCapabilities object, described in chapterB.3.6.2.

URL	
Pattern	{restfulprsdomain}/data/capabilities.{format}
Examples	http://www.prs.net/data/capabilities.xml http://www.prs.net/data/capabilities.json

Request variables			
format	**cardinality**	**value**	**description**
	0..1	json | xml | html	The format in which to return the metadata.

Methods	
Method	**Description**
GET	Retrieve capabilities.
HEAD,PUT,POST,DELETE	Not allowed.

Response Codes			
Method	**Code**	**Description**	**Returns**
GET	200 OK	Retrieve data capabilities	Headers, a DataCapabilities, see chapter 0, describing the capabilities of the data.

	404 Not Found	No operation with the specified URL exists.	Headers, An ErrorReport, see chapter 0
	406 Not Acceptable	The requested format or content-type cannot be delivered.	Headers, An ErrorReport, see chapter 0
HEAD,PUT, POST,DELETE	405 Method Not Allowed	Method not allowed.	Headers, An ErrorReport, see chapter 0

Response Headers			
Method	**Code**	**Header**	**Description**
GET	200 OK	Content-type	The mime type of the returned metadata.
HEAD,PUT, POST,DELETE	405 Not Allowed	Allow	The allowed methods: GET

B.5 Intended use cases

This chapter illustrates and describes some of the most common use cases for a RESTful portrayal registry.

B.5.1 Checking whether a Symbol exists or not

To make sure a Symbol exists before retrieving it, a HEAD request can be performed. If the Symbol exists a 200 OK will be returned. If not, a 404 Not Found will be returned. In the example illustrated in Figure 14, the Symbol exists and a 200 OK is returned.

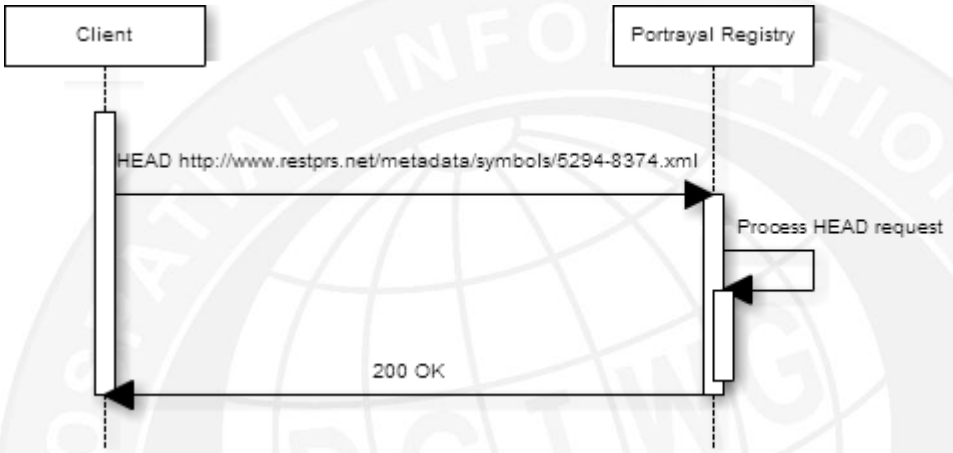


Figure 14: Check whether a Symbol exists or not

This method of checking whether an object exists is applicable to all object types in the registry.

B.5.2 Retrieving Symbol metadata

This is perhaps one of the most common use cases: retrieving the metadata for one object. In this case, the object is of the Symbol type. Only one piece of information is required to retrieve the metadata of an object; its identifier. As illustrated in Figure 15, the identifier in this case is “5294-8374”.

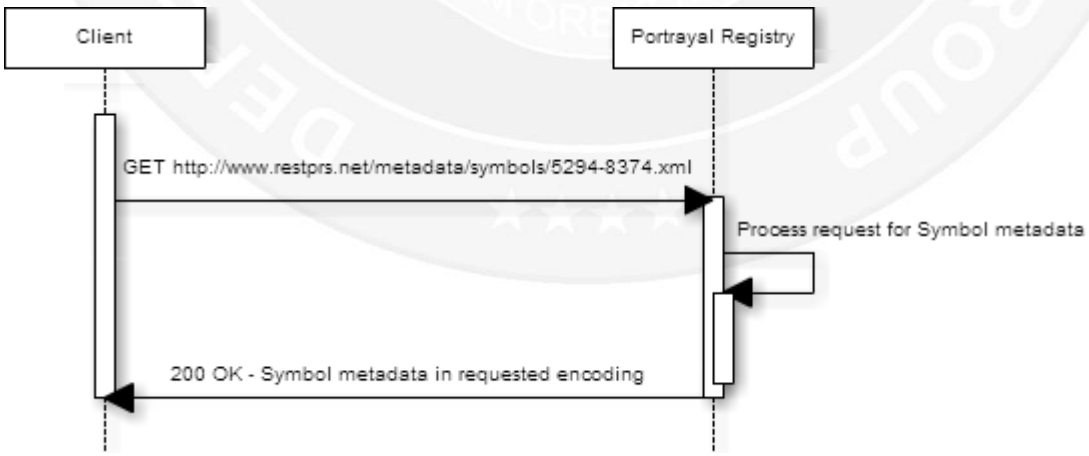


Figure 15: Retrieving Symbol metadata

B.5.3 Retrieving Symbol data

The process of retrieving Symbol data is very similar to that of retrieving Symbol metadata. The one difference from the client's perspective is the URL used to retrieve the resource. As illustrated in Figure 16, the data is accessed instead of the metadata.

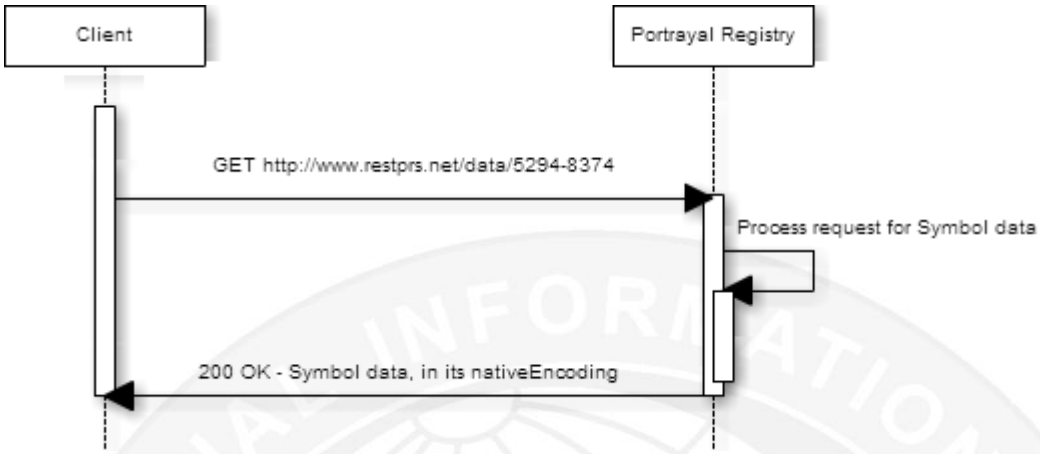


Figure 16: Retrieving Symbol data

B.5.4 Retrieving data for an entire RuleSet

When RuleSet data retrieval is supported, the client does not need to retrieve each Rule manually, but can instead use the identifier of the RuleSet when accessing the data, as illustrated in Figure 17.

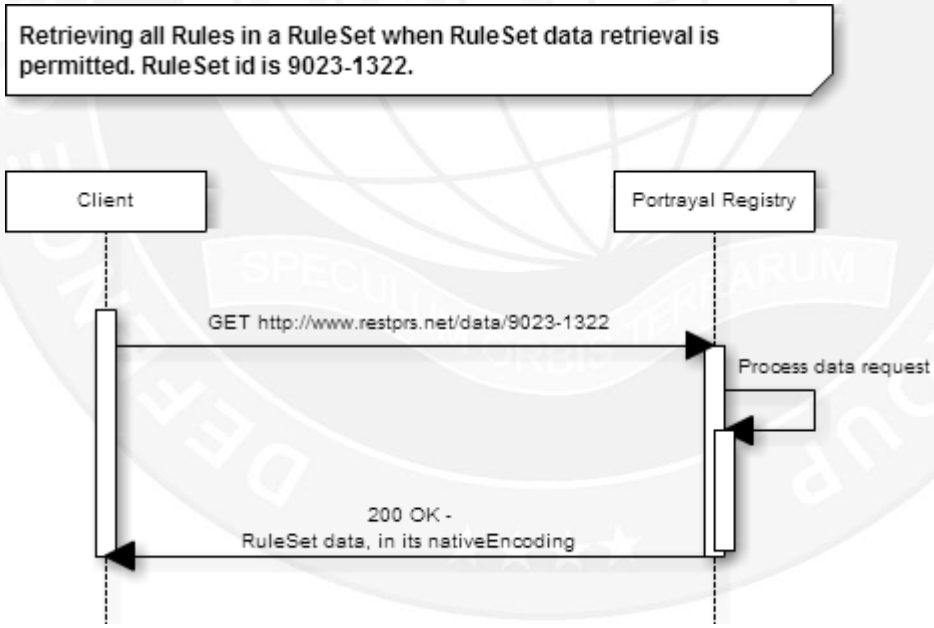


Figure 17: Retrieving RuleSet data

B.5.5 Retrieving data for all Rules in a RuleSet

When RuleSet data retrieval is not supported, the client will be responsible for retrieving each Rule in a RuleSet. This is performed by retrieving the RuleSet metadata and extracting all Rule identifiers from that metadata. Thereafter, for each Rule identifier, retrieve the data from the data - as illustrated in Figure 18.

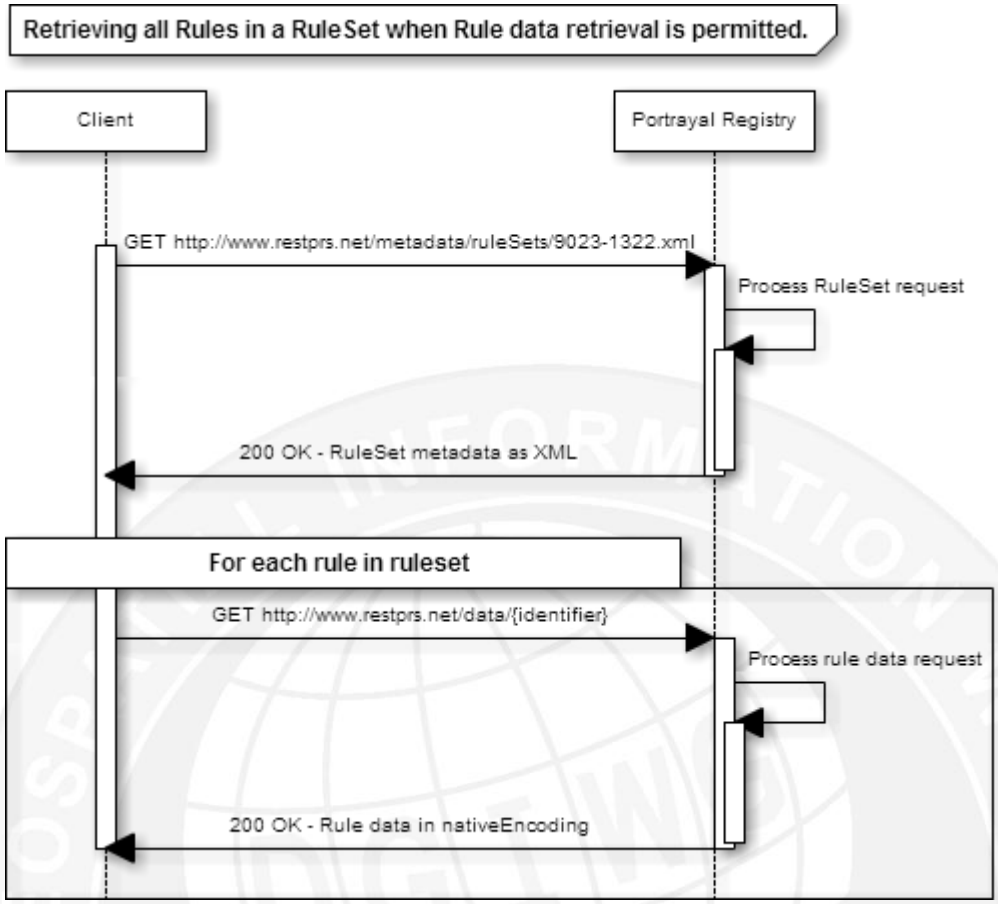


Figure 18: RuleSet data retrieval via RuleSet metadata

B.5.6 Retrieving RuleSets for a specific ApplicationSchema

In this use case, the client first performs a search and retrieves the metadata for all RuleSets matching a certain ApplicationSchema. Then, the client selects one of the RuleSets and performs data retrieval.

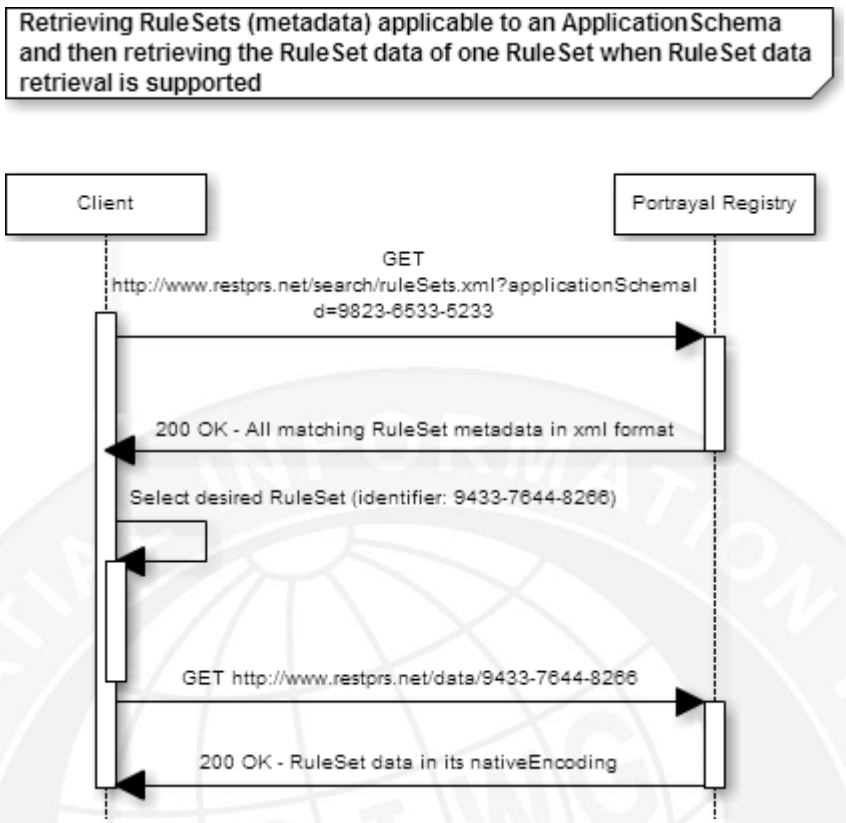


Figure 19: RuleSet data for RuleSet matching an ApplicationSchema

B.5.7 Using the PRS with an FPS

The client performs a GetMap request to the FPS and appends an SLD document containing links to the PRS expressed as OnlineResource references.

When the FPS receives the request it contacts the PRS, retrieves the portrayal rules and applies them when rendering map images. In this use case, only one Rule is referenced in the SLD and it is assumed that the PRS is capable of delivering Rule data. The use case is illustrated in Figure 20, below.

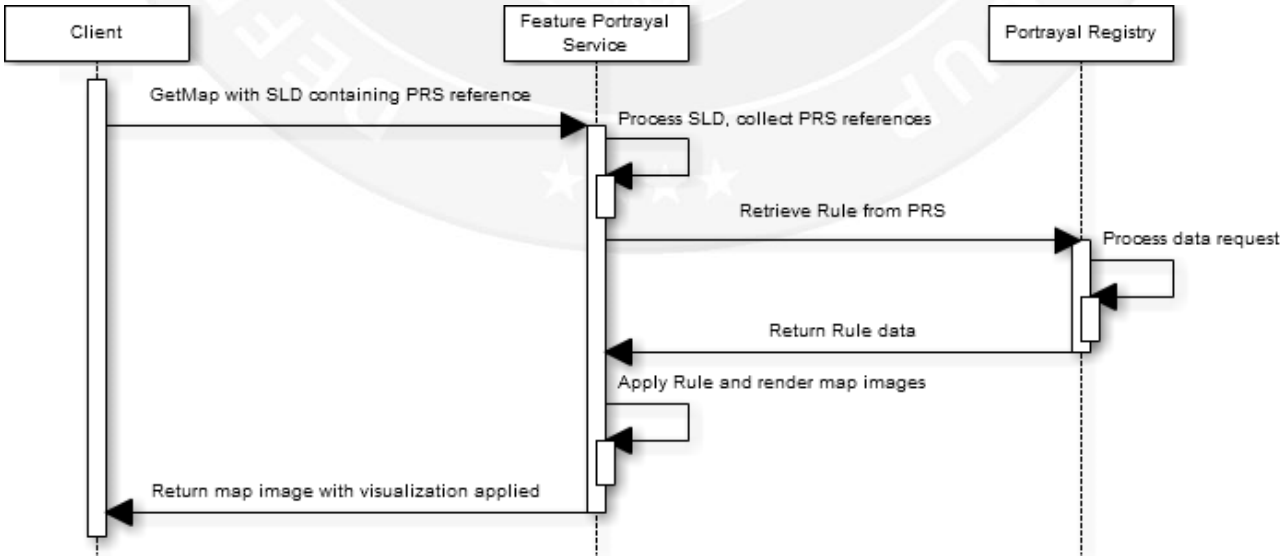


Figure 20: Using the PRS with a PRS-enabled FPS

B.6 XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://dgiwg.org/schemas/prs/rest"
    elementFormDefault="qualified"
    xmlns="http://dgiwg.org/schemas/prs/rest"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink">
  <!-- Imports -->
  <xs:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="http://www.w3.org/1999/xlink.xsd"/>

  <!-- Base objects -->
  <xs:complexType name="ItemBase">
    <xs:sequence>
      <xs:element name="Identifier" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="Name" type="xs:string" maxOccurs="1"/>
      <xs:element name="Description" type="xs:string" maxOccurs="1"/>
      <xs:element name="ResponsibleOrganization" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ResponseBase">
    <xs:sequence>
      <xs:element name="Timestamp" type="xs:dateTime" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <!-- References -->
  <xs:complexType name="ApplicationSchemaReferences">
    <xs:sequence>
      <xs:element name="Link" type="Link" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SymbolSetReferences">
    <xs:sequence>
      <xs:element name="Link" type="Link" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SymbolReferences">
    <xs:sequence>
      <xs:element name="Link" type="Link" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RuleSetReferences">
    <xs:sequence>
      <xs:element name="Link" type="Link" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RuleReferences">
    <xs:sequence>
      <xs:element name="Link" type="Link" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StandardFontReferences">
    <xs:sequence>
      <xs:element name="Link" type="Link" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StandardColorReferences">
    <xs:sequence>
      <xs:element name="Link" type="Link" maxOccurs="unbounded"/>
    </xs:sequence>
```

```
  </xs:complexType>

  <!-- Classifications-->
  <xs:simpleType name="SymbolClassification">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Complex"/>
      <xs:enumeration value="Line"/>
      <xs:enumeration value="Point"/>
      <xs:enumeration value="Polygon"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Cardinality">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Any"/>
      <xs:enumeration value="ZeroToOne"/>
      <xs:enumeration value="ExactlyOne"/>
      <xs:enumeration value="MoreThanZero"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Single Metadata item representations -->
  <xs:complexType name="Symbol">
    <xs:complexContent>
      <xs:extension base="ItemBase">
        <xs:sequence>
          <xs:element name="PreviewImage" type="Link" maxOccurs="1"/>
          <xs:element name="NativeEncoding" type="xs:string" maxOccurs="1"/>
          <xs:element name="IntendedUse" type="xs:string" maxOccurs="1"/>
          <xs:element name="AreaOfApplication" type="xs:string" maxOccurs="1"/>
          <xs:element name="SymbolType" type="SymbolClassification" />
          <xs:element name="SymbolSetReferences" type="SymbolSetReferences"
maxOccurs="1"></xs:element>
          <xs:element name="RuleReferences" type="RuleReferences"
maxOccurs="1"></xs:element>
          <xs:element name="StandardFontReferences" type="StandardFontReferences"
maxOccurs="1"></xs:element>
          <xs:element name="StandardColorReferences" type="StandardColorReferences"
maxOccurs="1"></xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="SymbolSet">
    <xs:complexContent>
      <xs:extension base="ItemBase">
        <xs:sequence>
          <xs:element name="SupportedEncodings" type="xs:string" />
          <xs:element name="SymbolReferences" type="SymbolReferences" minOccurs="1"
maxOccurs="1"></xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="Rule">
    <xs:complexContent>
      <xs:extension base="ItemBase">
        <xs:sequence>
          <xs:element name="NativeEncoding" type="xs:string" />
          <xs:element name="FeatureTypes" type="FeatureTypes" maxOccurs="1"/>
          <xs:element name="SymbolReferences" type="SymbolReferences" maxOccurs="1"/>
          <xs:element name="ApplicationSchemaReferences"
type="ApplicationSchemaReferences" maxOccurs="1"/>
```

```
          <xs:element name="RuleSetReferences" type="RuleSetReferences"
maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="RuleSet">
    <xs:complexContent>
      <xs:extension base="ItemBase">
        <xs:sequence>
          <xs:element name="SupportedEncodings" type="xs:string" />
          <xs:element name="PreviewImage" type="Link" maxOccurs="1"/>
          <xs:element name="RuleReferences" type="RuleReferences" minOccurs="1"
maxOccurs="1"></xs:element>
          <xs:element name="ApplicationSchemaReferences"
type="ApplicationSchemaReferences" maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ApplicationSchema">
    <xs:complexContent>
      <xs:extension base="ItemBase">
        <xs:sequence>
          <xs:element name="FeatureTypes" type="FeatureTypes" maxOccurs="1"/>
          <xs:element name="RuleReferences" type="RuleReferences"
maxOccurs="1"></xs:element>
          <xs:element name="RuleSetReferences" type="RuleSetReferences"
maxOccurs="1"></xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="StandardFont">
    <xs:complexContent>
      <xs:extension base="ItemBase">
        <xs:sequence>
          <xs:element name="PreviewImage" type="Link" maxOccurs="1"/>
          <xs:element name="SymbolReferences" type="SymbolReferences"
maxOccurs="1"></xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="StandardColor">
    <xs:complexContent>
      <xs:extension base="ItemBase">
        <xs:sequence>
          <xs:element name="PreviewImage" type="Link" maxOccurs="1"/>
          <xs:element name="SymbolReferences" type="SymbolReferences"
maxOccurs="1"></xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Multiple metadata item representations (used in collections) -->
  <xs:complexType name="Symbols">
    <xs:sequence>
      <xs:element name="Symbol" type="Symbol" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>
```

```
  <xs:complexType name="SymbolSets">
    <xs:sequence>
      <xs:element name="SymbolSet" type="SymbolSet"
maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Rules">
    <xs:sequence>
      <xs:element name="Rule" type="Rule" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RuleSets">
    <xs:sequence>
      <xs:element name="RuleSet" type="RuleSet" maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ApplicationSchemas">
    <xs:sequence>
      <xs:element name="ApplicationSchema" type="ApplicationSchema"
maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StandardFonts">
    <xs:sequence>
      <xs:element name="StandardFont" type="StandardFont"
maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StandardColors">
    <xs:sequence>
      <xs:element name="StandardColor" type="StandardColor"
maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>

  <!-- Collections -->
  <xs:complexType name="SymbolCollection">
      <xs:sequence>
        <xs:element name="Symbols" type="Symbols" maxOccurs="1"/>
      </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SymbolSetCollection">
      <xs:sequence>
        <xs:element name="SymbolSets" type="SymbolSets" maxOccurs="1"/>
      </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RuleCollection">
      <xs:sequence>
        <xs:element name="Rules" type="Rules" maxOccurs="1"/>
      </xs:sequence>
  </xs:complexType>
  <xs:complexType name="RuleSetCollection">
      <xs:sequence>
        <xs:element name="RuleSets" type="RuleSets" maxOccurs="1"/>
      </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ApplicationSchemaCollection">
    <xs:sequence>
      <xs:element name="ApplicationSchemas" type="ApplicationSchemas" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StandardFontCollection">
```

```
      <xs:sequence>
        <xs:element name="StandardFonts" type="StandardFonts" maxOccurs="1"/>
      </xs:sequence>
  </xs:complexType>
  <xs:complexType name="StandardColorCollection">
    <xs:sequence>
      <xs:element name="StandardColors" type="StandardColors" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="FeatureTypes">
    <xs:sequence>
      <xs:element name="FeatureType" type="xs:string"
maxOccurs="unbounded"></xs:element>
    </xs:sequence>
  </xs:complexType>

  <!-- Responses and related types-->
  <xs:complexType name="MetadataResponse">
    <xs:complexContent>
      <xs:extension base="ResponseBase">
        <xs:sequence>
          <xs:element name="ResultCount" type="xs:int" maxOccurs="1"></xs:element>
          <xs:choice maxOccurs="1">
            <xs:element name="Symbol" type="Symbol"></xs:element>
            <xs:element name="SymbolSet" type="SymbolSet"></xs:element>
            <xs:element name="Rule" type="Rule"></xs:element>
            <xs:element name="RuleSet" type="RuleSet"></xs:element>
            <xs:element name="ApplicationSchema"
type="ApplicationSchema"></xs:element>
            <xs:element name="StandardFont" type="StandardFont"></xs:element>
            <xs:element name="StandardColor" type="StandardColor"></xs:element>
          </xs:choice>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="SearchResponse">
    <xs:complexContent>
      <xs:extension base="ResponseBase">
        <xs:sequence>
          <xs:element name="ResultCount" type="xs:int" maxOccurs="1"></xs:element>
          <xs:choice maxOccurs="1">
            <xs:element name="SymbolCollection" type="SymbolCollection"></xs:element>
            <xs:element name="SymbolSetCollection"
type="SymbolSetCollection"></xs:element>
            <xs:element name="RuleCollection" type="RuleCollection"></xs:element>
            <xs:element name="RuleSetCollection"
type="RuleSetCollection"></xs:element>
            <xs:element name="ApplicationSchemaCollection"
type="ApplicationSchemaCollection"></xs:element>
            <xs:element name="StandardFontCollection"
type="StandardFontCollection"></xs:element>
            <xs:element name="StandardColorCollection"
type="StandardColorCollection"></xs:element>
          </xs:choice>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="ErrorReport">
    <xs:complexContent>
```

```
      <xs:extension base="ResponseBase">
        <xs:sequence>
          <xs:element name="ErrorTitle" type="xs:string" minOccurs="1" maxOccurs="1"/>
          <xs:element name="ErrorDescription" type="xs:string" minOccurs="1"
maxOccurs="1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="OnlineResource">
    <xs:sequence>
      <xs:element name="Link" type="Link"/>
      <xs:element name="ContentTypes" type="ContentTypes" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OnlineResources">
    <xs:sequence>
      <xs:element name="OnlineResource" type="OnlineResource" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ContentTypes">
    <xs:sequence>
      <xs:element name="ContentType" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Capabilities -->
  <xs:complexType name="Link">
    <xs:attributeGroup ref="xlink:simpleAttrs"/>
  </xs:complexType>
  <xs:complexType name="Endpoints">
    <xs:sequence>
      <xs:element name="StandardColor" type="OnlineResource"
maxOccurs="1"></xs:element>
      <xs:element name="StandardFont" type="OnlineResource"
maxOccurs="1"></xs:element>
      <xs:element name="SymbolSet" type="OnlineResource" maxOccurs="1"></xs:element>
      <xs:element name="Symbol" type="OnlineResource" maxOccurs="1"></xs:element>
      <xs:element name="RuleSet" type="OnlineResource" maxOccurs="1"></xs:element>
      <xs:element name="Rule" type="OnlineResource" maxOccurs="1"></xs:element>
      <xs:element name="ApplicationSchema" type="OnlineResource"
maxOccurs="1"></xs:element>
      <xs:element name="PreviewImages" type="OnlineResource"
maxOccurs="1"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Capabilities">
    <xs:complexContent>
      <xs:extension base="ResponseBase">
        <xs:sequence>
          <xs:element name="Title" type="xs:string" maxOccurs="1"></xs:element>
          <xs:element name="Description" type="xs:string" maxOccurs="1"></xs:element>
          <xs:element name="Endpoints" type="Endpoints" maxOccurs="1"></xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="DataCapabilities">
    <xs:complexContent>
      <xs:extension base="Capabilities">
      </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="MetadataCapabilities">
    <xs:complexContent>
      <xs:extension base="Capabilities">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!--SearchResponse-->
  <xs:complexType name="Operations">
    <xs:sequence>
      <xs:element name="GetStandardColor" type="OnlineResource"
maxOccurs="1"></xs:element>
      <xs:element name="GetStandardFont" type="OnlineResource"
maxOccurs="1"></xs:element>
      <xs:element name="GetSymbolSet" type="OnlineResource"
maxOccurs="1"></xs:element>
      <xs:element name="GetSymbol" type="OnlineResource" maxOccurs="1"></xs:element>
      <xs:element name="GetRuleSet" type="OnlineResource" maxOccurs="1"></xs:element>
      <xs:element name="GetRule" type="OnlineResource" maxOccurs="1"></xs:element>
      <xs:element name="GetApplicationSchema" type="OnlineResource"
maxOccurs="1"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="SearchCapabilities">
    <xs:complexContent>
      <xs:extension base="ResponseBase">
        <xs:sequence>
          <xs:element name="Title" type="xs:string" maxOccurs="1"></xs:element>
          <xs:element name="Description" type="xs:string" maxOccurs="1"></xs:element>
          <xs:element name="Link" type="Link" maxOccurs="1"/>
          <xs:element name="Operations" type="Operations" maxOccurs="1"></xs:element>
          <xs:element name="SubServices" type="OnlineResources"
maxOccurs="1"></xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

B.7 JSON Schema

Each subchapter holds the content of one schema file.

B.7.1 ApplicationSchema

```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/applicationSchema.js",
	"title":"applicationSchema",
	"type":"object",
	"extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
	"properties":{
		"featureTypes":{
			"title":"featureTypes",
			"description":"A uri pointing to a preview image of a visualization
using this symbol",
			"type":"array",
			"items":{
				"title":"featureType",
				"type":"string"}
		},
		"ruleReferences":{
			"description":"Rules referencing this appSchema",
		"type":"array",
		"items":{
				"type":"string","format":"uri"}
		},
		"ruleSetReferences":{
			"description":"RuleSets referencing this appSchema",
		"type":"array",
			"items":{
			   "type":"string","format":"uri"}
		}
	}
}
```

B.7.2 Cardinality

```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/cardinality.js",
	"title":"cardinality",
	"description":"Defines a cardinality",
	"type":"object",
    "properties":{
		"cardinality":{
		"default":"Any",
		  "enum":["Any","ZeroToOne","ExactlyOne","MoreThanZero"],
			"type":"string"
		}
    }
}
```

B.7.3 Endpoints


```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/endpoints.js",
	"title":"endpoints",
	"type":"object",
	"properties":{
		"standardColor":{
			"description":"available endpoints from which to retrieve
standardColors",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"standardFont":{
			"description":"available endpoints from which to retrieve
standardFonts",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"symbolSet":{
			"description":"available endpoints from which to retrieve
symbolSets",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"symbol":{
			"description":"available endpoints from which to retrieve symbols",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"ruleSet":{
			"description":"available endpoints from which to retrieve ruleSets",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"rule":{
			"description":"available endpoints from which to retrieve rules",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"applicationSchema":{
			"description":"available endpoints from which to retrieve
applicationSchema",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"previewImage":{
			"description":"available endpoints from which to retrieve
previewImages",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		}
	}
}
```

B.7.4 ErrorReport


```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/errorReport.js",
	"title":"errorReport",
	"type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/responseBase.js"},
	"properties":{
		"errorTitle":{
			"description":"A descriptive error title",
			"required":true,
			"type":"string"
		},
		"errorDescription":{
			"description":"A detailed description of the error and measures that
can be taken, if any exist, to avoid the error",
			"required":true,
			"type":"string"
		}
	}
}
```

B.7.5 ItemBase


```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/itemBase.js",
	"title":"itemBase",
	"type":"object",
	"properties":{
		"identifier":{
			"title":"Unique identifier",
			"description":"A unique identifier for the object",
			"required":true,
			"type":"string"
		},
		"name":{
			"title":"Object name",
			"description":"A human readable name",
		"required":false,
			"type":"string"
		},
		"description":{
			"title":"Object description",
			"description":"A human readable description of the object and/or its
contents",
		"required":false,
			"type":"string"
		},
		"responsibleOrganization":{
			"title":"Responsible organization",
		"required":true,
			"description":"The name of the organization responsible for creation
and/or maintenance of the object",
			"type":"string"
		}
	}
}
```

B.7.6 OnlineResource

```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/onlineResource.js",
	"title":"onlineResource",
	"type":"object",
	"properties":{
		"link":{
			"title":"resource link",
			"type":"string",
			"format":"uri"
		},
		"contentTypes":{
			"description":"a list of content types (mime types) that can be
retrieved from the onlineResource",
			"type":"array",
			"items":{
				"title":"contentType",
				"description":"a mime type",
				"type":"string"
	}
	  }
	}
}
```

B.7.7 Operations

```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/operations.js",
	"title":"endpoints",
	"type":"object",
	"properties":{
		"getStandardColor":{
			"description":"available endpoints from which to search for
standardColors",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"getStandardFont":{
			"description":"available endpoints from which to search for
standardFonts",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"getSymbolSet":{
			"description":"available endpoints from which to search for
symbolSets",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"getSymbol":{
			"description":"available endpoints from which to search for symbols",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"getRuleSet":{
			"description":"available endpoints from which to search for
ruleSets",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"getRule":{
			"description":"available endpoints from which to search for rules",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		},
		"getApplicationSchema":{
			"description":"available endpoints from which to search for
applicationSchema",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResource.js"
		}
	}
}
```

B.7.8 MetadataCapabilities

```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/metadataCapabilities.js",
       "title":"metadataCapabilities",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/responseBase.js"},
       "properties":{
              "name":{
            "required":false,
                     "description":"A descriptive name",
                     "type":"string"
              },
              "description":{
            "required":false,
                     "description":"A human readable description of the returned
capabilities",
                     "type":"string"
              },
              "endpoints":{
                     "description":"The available endpoints of the metadata",
                     "$ref":"http://dgiwg.org/schemas/prs/rest/endpoints.js"
              }
       }
}
```

B.7.9 MetadataResponse

```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/metadataResponse.js",
       "title":"metadataResponse",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/responseBase.js"},
       "properties":{
              "resultItem":{
                     "description":"The metadata for the item with the matching
identifier. At most one object is returned",
                     "type":"object"
              },
              "resultCount":{
                     "title":"resultCount",
                     "required":true,
                     "type":"integer",
                     "default":0
              }
       }
}
```

B.7.10 DataCapabilities

```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/dataCapabilities.js",
	"title":"dataCapabilities",
	"type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/responseBase.js"},
	"properties":{
		"name":{
			"description":"A descriptive name",
			"required":false,
			"type":"string"
		},
		"description":{
			"description":"A human readable description of the returned
capabilities",
            "required":false,
			"type":"string"
		},
		"endpoints":{
			"description":"The available endpoints of the data",
			"$ref":"http://dgiwg.org/schemas/prs/rest/endpoints.js"
		}
	}
}
```

B.7.11 ResponseBase

```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/responseBase.js",
	"title":"responseBase",
	"type":"object",
	"properties":{
		"timestamp":{
			"title":"Timestamp",
			"description":"A timestamp denoting the time the response was sent",
			"required":true,
			"type":"string",
			"format":"date-time"
		}
	}
}
```

B.7.12 Rule


```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/rule.js",
       "title":"rule",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
       "properties":{
              "nativeEncoding":{
                     "title":"Native Encoding",
                     "description":"The native encoding of this symbol",
                     "type":"string"
              },
              "featureTypes":{
                     "title":"Feature Types",
                     "description":"A collection of feature type names",
                     "type":"array",
                     "items":{
                            "title":"featureType",
                            "description":"A featureTypeName",
                            "type":"string"
                     }
              },
              "symbolReferences":{
                     "description":"All symbols referenced by this rule",
            "type":"array",
                     "items":{
                        "type":"string","format":"uri"}
              },
              "applicationSchemaReferences":{
                     "description":"All application schema referenced by this rule",
            "type":"array",
                     "items":{
                        "type":"string","format":"uri"}
              },
              "ruleSetReferences":{
                     "description":"All rule sets in which this rule is used",
            "type":"array",
                     "items":{
                        "type":"string","format":"uri"}
              }
       }
}
```

B.7.13 RuleSet


```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/ruleSet.js",
       "title":"ruleSet",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
       "properties":{
              "supportedEncodings":{
                     "title":"Supported Encodings",
            "required":true,
                     "description":"The supported encodings of this rule set",
                     "type":"string"
              },
              "previewImage":{
                     "title":"PreviewImage",
            "required":false,
                     "description":"A uri pointing to a preview image of a visualization
using this rule set",
                     "type":"string",
                     "format":"uri"
              },
              "ruleReferences":{
                     "description":"Rules referenced by this Rule Set",
            "required":true,
            "type":"array",
                     "items":{
                        "type":"string","format":"uri"}
              }
       }
}
```

B.7.14 SearchResponse


```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/searchResponse.js",
       "title":"searchResponse",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
       "properties":{
              "resultItems":{
                     "description":"The metadata for the registry items matching the
search.",
                     "type":"array",
            "minItems":0,
                     "items":{
                            "title":"item",
                            "description":"each result is an object of the type that was
searched for.",
                            "type":"object"}
              },
              "resultCount":{
                     "title":"resultCount",
                     "required":true,
                     "type":"integer"
              }
       }
}
```

B.7.15 SearchCapabilities


```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/searchCapabilities.js",
	"title":"searchCapabilities",
	"type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/responseBase.js"},
	"properties":{
		"title":{
			"description":"A descriptive title",
			"required":false,
			"type":"string"
		},
		"description":{
			"description":"A human readable description of the returned
capabilities",
            "required":false,
			"type":"string"
		},
		"link":{
			"description":"A link to the service",
			"type":"string",
			"format":"uri"
		},
		"operations":{
			"description":"The available operations of the service",
			"$ref":"http://dgiwg.org/schemas/prs/rest/operations.js"
		},
		"subServices":{
			"description":"The available sub services of the service",
			"$ref":"http://dgiwg.org/schemas/prs/rest/onlineResources.js"
		}
	}
}
```

B.7.16 StandardColor

```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/standardColor.js",
       "title":"standardColor",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
       "properties":{
              "previewImage":{
                     "title":"Preview image",
                     "description":"A uri pointing to a preview image of a visualization
using this color",
                     "type":"string",
                     "format":"uri"
              },
              "symbolReferences":{
                     "description":"Symbols referencing this StandardColor",
                     "items":{
                         "type":"string","format":"uri"}
              }
       }
}
```

B.7.17 StandardFont

```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/standardFont.js",
       "title":"standardFont",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
       "properties":{
              "previewImage":{
                     "title":"Preview image",
                     "description":"A uri pointing to a preview image of a visualization
using this font",
                     "type":"string",
                     "format":"uri"
              },
              "symbolReferences":{
                     "description":"Symbols referencing this StandardFont",
            "type":"array",
                     "items":{
                         "type":"string","format":"uri"}
              }
       }
}
```

B.7.18 Symbol


```
{
       "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
       "id":"http://dgiwg.org/schemas/prs/rest/symbol.js",
       "title":"symbol",
       "type":"object",
    "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
       "properties":{
              "nativeEncoding":{
                     "title":"Native Encoding",
                     "description":"The native encoding of this symbol",
                     "type":"string"
              },
        "symbolClassification":{
            "$ref":"http://dgiwg.org/schemas/prs/rest/symbolClassification.js"
        },
              "intendedUse":{
                     "title":"Intended use",
                     "description":"The intended use of this symbol",
                     "type":"string"
              },
              "areaOfApplication":{
                     "title":"Area Of Application",
                     "description":"The area of application for this symbol, e.g.
'tactical' or 'nautical'",
                     "type":"string"
              },
              "previewImage":{
                     "title":"PreviewImage",
                     "description":"A uri pointing to a preview image of a visualization
using this symbol",
                     "type":"string",
                     "format":"uri"
              },
              "symbolSetReferences":{
                     "description":"A list of Links representing symbolSets in which this
symbol is used",
           "type":"array",
                     "items":{
                         "type":"string","format":"uri"}
              },
              "ruleReferences":{
                     "description":"A list of Links representing rules in which this
symbol is used",
                     "type":"array",
                     "items":{
                         "type":"string","format":"uri"}
              },
              "standardFontReferences":{
                     "description":"A list of Links representing standardFonts which this
symbol uses",
                     "type":"array",
                     "items":{
                         "type":"string","format":"uri"}
              },
              "standardColorReferences":{
                     "description":"A list of Links representing standardColors which this
symbol uses",
                     "type":"array",
```

B.7.19 SymbolClassification


```
{
	"$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
	"id":"http://dgiwg.org/schemas/prs/rest/symbolClassification.js",
	"title":"symbolClassification",
	"type":"string",
	"description":"Defines what type of geographical object a symbol is meant to
portray",
	"required":true,
	"enum":["Complex","Line","Point","Polygon"]
	}
}
```


B.7.20 SymbolSet


```
   {
      "$schema":"http://tools.ietf.org/id/draft-zyp-json-schema-03.html#",
      "id":"http://dgiwg.org/schemas/prs/rest/symbolSet.js",
      "title":"symbolSet",
      "type":"object",
     "extends":{"$ref":"http://dgiwg.org/schemas/prs/rest/itemBase.js"},
      "properties":{
            "supportedEncodings":{
                  "title":"Supported Encodings",
                  "description":"The supported encodings of this symbol set",
                  "type":"string"
            },
            "symbolReferences":{
                  "description":"Symbols referenced by this Symbol Set",
          "type":"array",
                  "items":{
                     "type":"string","format":"uri"}
            }
      }
   }
```

B.8 REST interface Abstract Test Suite

B.8.1 Portrayal registry service

B.8.1.1 Mandatory metadata output formats

a) Test purpose: Verify that the Portrayal Registry Service supports all metadata output formats mandated in this annex.

b) Test Method: Retrieve the capabilities documents for the metadata and search endpoints. Verify that the ContentTypes elements of all individual Operations and Endpoints include all mandated content types.

c) References: B.4.5.8 (metadata capabilities request), B.4.6.8 (search capabilities request), B.4.2 (mandatory metadata output formats).

d) Test type: Basic

B.8.1.2 Mandatory metadata endpoints

a) Test purpose: Verify that the Portrayal Registry Service supports all metadata endpoints mandated in this annex.

b) Test Method: Retrieve the capabilities document for the metadata endpoint. Verify that the service provides Links for all Endpoints in the document.

c) References: B.4.5.8 (metadata capabilities request), B.3.6.1 (MetadataCapabilities format).

d) Test type: Basic

B.8.1.3 Mandatory search operations

a) Test purpose: Verify that the Portrayal Registry Service supports all search operations mandated in this annex.

b) Test Method: Retrieve the capabilities document for the search endpoint. Verify that the service provides Links for all Operations in the document.

c) References: B.4.6.8 (search capabilities request), B.3.6.3 (SearchCapabilities format).

d) Test type: Basic

B.8.1.4 Valid XML output

a) Test purpose: Verify that the XML output of the Portrayal Registry Service conforms to the XML schema specified in this annex.

b) Test Method: Retrieve objects in XML format from each of the Endpoints in the metadata resource. Validate the returned objects against the XML schema.

c) References: B.6 (XML schema).

d) Test type: Basic

B.8.1.5 Valid JSON output

a) Test purpose: Verify that the JSON output of the Portrayal Registry Service conforms to the XML schema specified in this annex.

b) Test Method: Retrieve objects in JSON format from each of the Endpoints in the metadata resource. Validate the returned objects against the JSON schema.

c) References: B.7 (JSON schema).

d) Test type: Basic

B.8.1.6 Information model referential integrity

a) Test purpose: Verify that objects output from the Portrayal Registry Service contain correct references to other objects in the registry.

b) Test Method: Retrieve objects of each class from the metadata endpoint. Verify that it is possible to retrieve all objects that are referenced in Link elements in the object representations.

c) References: B.3.1.4, B.3.1.5, B.3.1.6, B.3.1.7, B.3.1.8, B.3.1.9 (individual object representations).

d) Test type: Basic

Bibliography

[1] IETF URI Template
http://tools.ietf.org/html/draft-gregorio-uritemplate-08#section-1.2

[2] Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.
http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[3] ebXML Registry Services and Protocols - Version 3.0 – 2 May, 2005
http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.pdf

[4] Styled Layer Descriptor profile of the Web Map Service, version 1.1.0 (2007-06-29), Dr. Markus Lupp, available at *http://www.opengeospatial.org/standards/sld*

[5] Styled Layer Descriptor profile of the Web Map Service, version 1.0 (2002-09-19), William Lalonde, available at *http://www.opengeospatial.org/standards/sld*

[6] Symbology Encoding, version 1.1.0 (2006-07-21), Dr. Markus Müller, available at *http://www.opengeospatial.org/standards/se*

[7] OGC KML, version 2.2.0 (2008-04-14), Tim Wilson, available at *http://www.opengeospatial.org/standards/kml*

[8] ISO 19135:2005, Geographic information -- Procedures for item registration, available at *http://www.iso.org/*