MyData Service Registry Specification

Notice

MyData Architecture defines the operations and APIs between the Operational Roles (Operator, Source, Sink etc.). Any descriptions or figures of the role's internal structure or operations are for illustrative purposes only.

1. Introduction

This document specifies Service Registry, a part of MyData Operator.

This document is part of the MyData architecture release 1.1. It is assumed that the reader is familiar with MyData Architecture - Consent Based Approach for Personal Data Management document available at https://hiit.github.io/mydata-stack/.

Known deficiencies in this release: missing JSON structures, API naming, message formats, and error messages. These will be part of the next release of this document.

1.1 Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2 Terminology

Key terminology used in this specification is defined in the Glossary of MyData Architecture - Consent Based Approach for Personal Data Management release 1.1 available at https://hiit.github.io/mydata-stack/.

1.3 Service Registry

Service Registry is part of the MyData Operator and it provides two major functions: it maintains a database of all the services accessible with this Operator (Service Registration) and it enable searching for compatible services (Service Discovery).

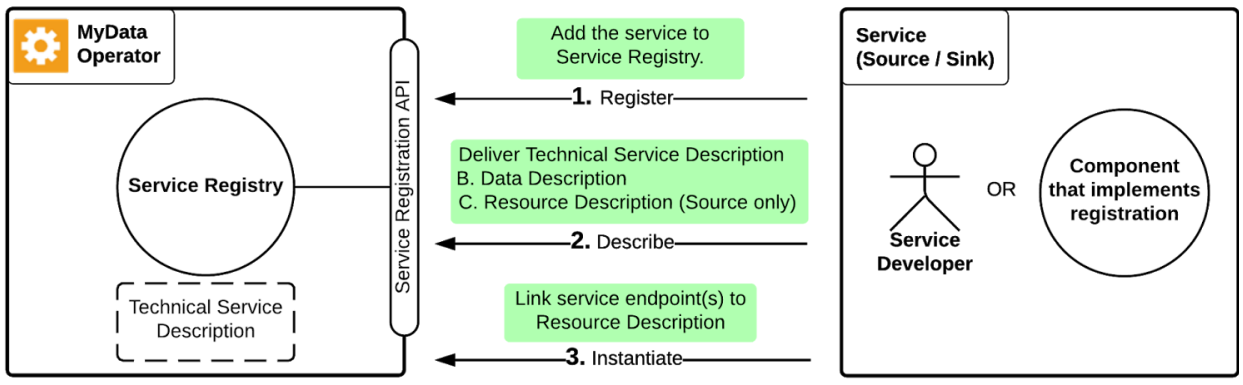

Figure 1.1. Registering a service to a Service Registry

Figure 1.1 presents the phases of Service Registration. Service provider registers the services and receives a unique ID (within this registry) for the service (1), create service descriptions. Service Registry provides identities to all services registered to an Operator and provides access to Service Descriptions (2). A service instance is an entity that implements the described service interfaces (3).

The Service Registry also enables service developers to discover and integrate compatible Sources and Sinks. This is supported by a Service Description that consists of multiple parts. A Service Data Description enables discovery of compatible services and supports interoperability between services. It describes data provided by a Source or required by a Sink. A Technical Service Description is a machine-readable API description of the service interface and is essential for service developers for enabling service integration.

Service Registry source code is available at https://github.com/digitalhealthrevolution/serviceregistry and original reference implementation of the Service Registry can be found at www.digitalserviceshub.com. This document gives an overview of the Service Registry data model and functionalities. As an enhancement to the open source version, it defines a more detailed Service Data Description.

2. Service Registry

The main feature of Service Registry is to allow registering and discovering available services and, thus, facilitate service developers and service providers to manage registrations and discover the services. Service Registry is a component that provides identities to all services registered to an Operator. Each service also has a comprehensive Service Description accessible through the Service Registry. A Service Description consists of multiple parts enabling different types of service discovery.

Two types of MyData services have been identified. Services providing data resources to others are called Source services and services requiring data from other services are called Sink services. The main difference for these two service types is that the Sink services do not expose a service interface for data access and the Source services do.

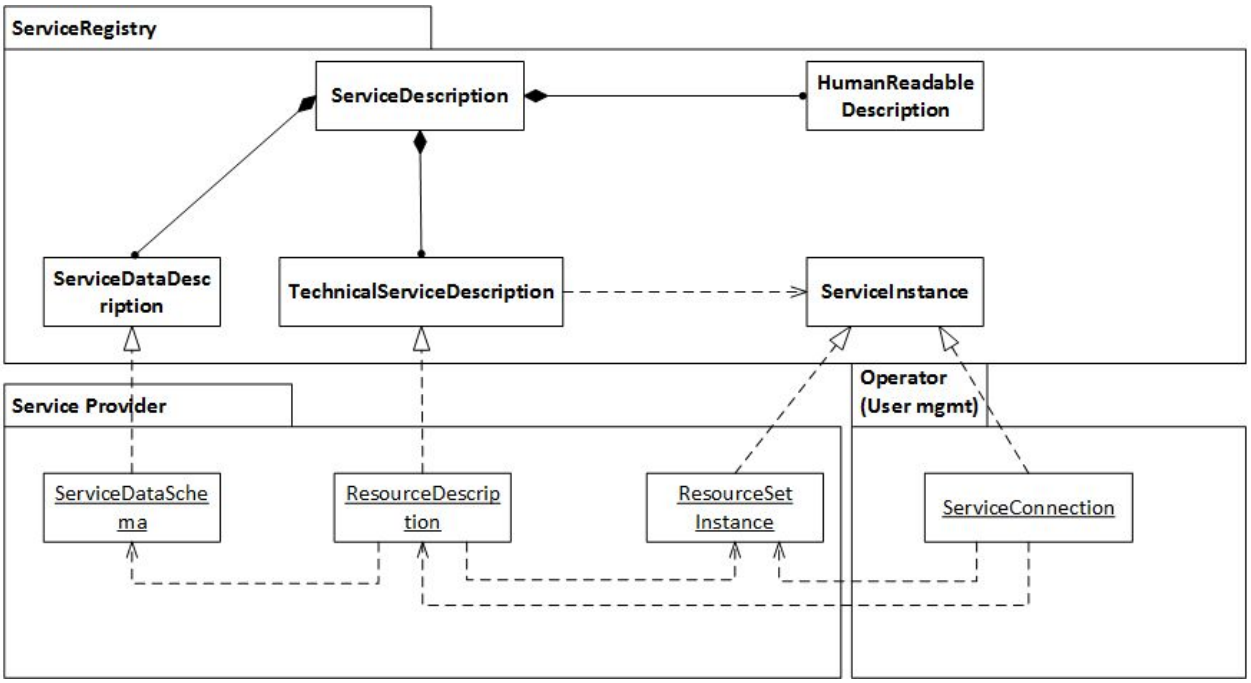

Figure 2.1. Realization of a service registration

For each service, a Service Description is required. The Service Description consists of multiple parts that serve different purposes. For end-users, a Human Readable Description of the service and its purpose etc. is mandatory. In addition to this, several Technical Descriptions on the service are required. For source type services, a set of data resource descriptions can be stored in the registry. For service matching (finding relevant Source services providing data for Sink services), it is preferred that for both, Sink and Source services the data structure is described as ServiceDataDescriptions. In addition to providing the identity and description of services, the Service Registry also manages the endpoints of the registered services (ServiceAccessURI).

This Service Registry implementation is based on the VTT DigitalServicesHub Service Registry implementation, which source code is available at https://github.com/digitalserviceshub/serviceregistry. The MyData Service Description requirements have been mapped to the original Service Registry data model and the data model has been extended by introduction of the Service Data Description -concept (Figure 2.1). This implementation is available at ttps://github.com/digitalhealthrevolution/serviceregistry.

2.1 Service registration

Service registration is a process, where the necessary information for using and discovering the service are published in the registry in a uniform way. First, a description of a service is generated in the registry. The parts of the service description are described in the previous chapters. For each Source service, at least one technical interface description is required (e.g. WADL document for REST interface). For enabling advanced service matching, service data descriptions are also enforced. In addition to these technical documents, a set of variables describing the service to end users (e.g. name, version number, icon, human readable description, …) and identifying the service (e.g. provider information) are also registered. A service developer forms the required service description documents and registers the description to the registry. These documents are stored on the service provider's server and only referenced through the formed resources from the Service Registry.

ServiceRegistry root resource for creating service descriptions utilized in the Service Registry.

Operations

Method	Params [type, name, notes]	Return values	Notes
GET()		ServiceRegistryEntry[]	
POST()	ServiceRegistryEntry newEntry	boolean	requires HTTP Basic auth credentials

Figure 2.2 presents the service registration process and the formed resources for each service. The Service Registry data model is presented in chapter 2.2 and details about Service Registry ROA (Resource-Oriented Architecture) style service interfaces can be found at Appendix 1 - Service Registry ROA Interface.

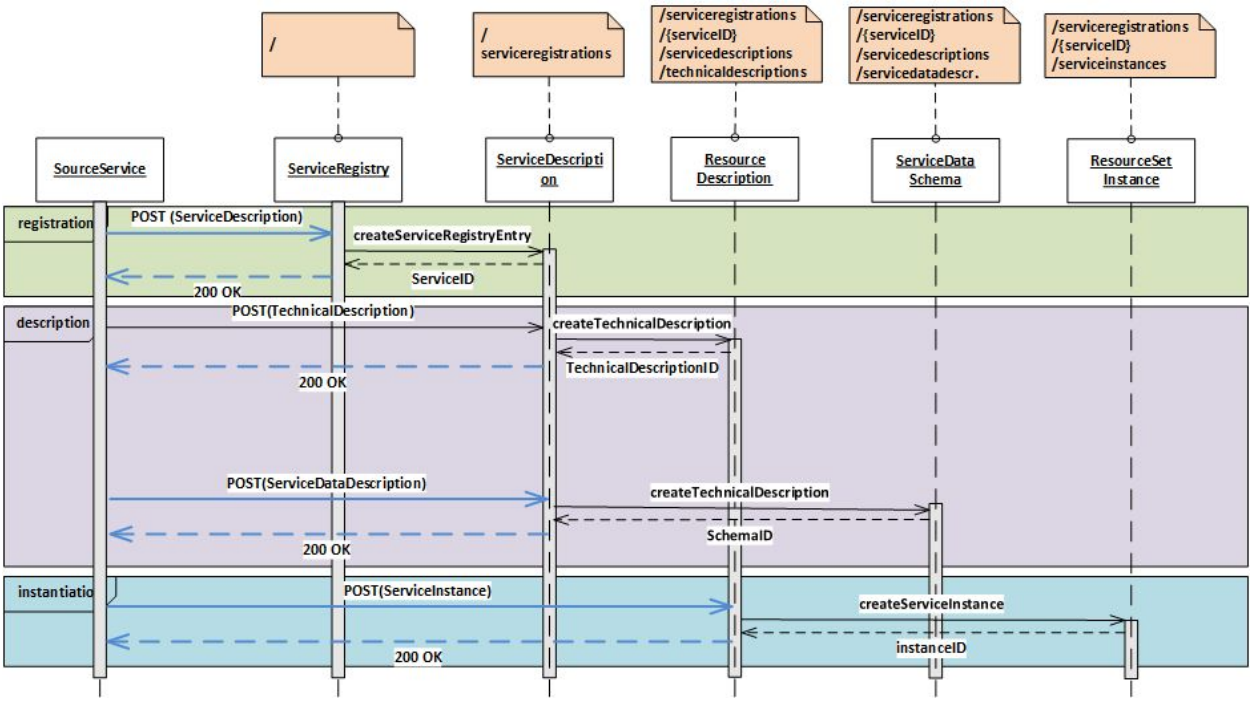

Figure 2.2. The service registration process and the formed resources for each service

Registration

The first part of the service registration process is registration of the Service Description. For each MyData service, sink or source, a Service Description including the basic information on the service and its author is required. For each registered service, a serviceRegistryEntry is created that contains the serviceID used for managing this particular service. A ServiceDescription resource is created for the service that is used for managing the technical descriptions of the service. The Service registration is managed by using the /serviceregistrations/{serviceid} resource.

ServiceRegistrations resource for updating, retrieving and removing service descriptions utilized in the Service Registry:

Operations

Method	Params [type, name, notes]	Return values	Notes
DELETE()	string id	boolean	requires HTTP Basic auth credentials
GET()	string id	ServiceDescription	
PUT()	ServiceDescription newDescription	boolean	requires HTTP Basic auth credentials

Description

For each MyData service, a set of technical (machine readable, e.g. WADL) descriptions are required or enforced in addition to the basic registration and description. In the registration phase, the needed resources for managing the Technical Service Descriptions have been created.

For each source service, a ResourceDescription describing the service API is required. In addition to these, for both sink and source services, ServiceDataDescriptions are enforced for enabling intelligent service matching for end users.

ServiceRegistrations/{service_id}/servicedescriptions/technicaldescriptions resource for updating, retrieving and removing technical service descriptions utilized in the Service Registry:

Operations

Method	Params [type, name, notes]	Return values	Notes
DELETE()	string id	boolean	requires HTTP Basic auth credentials
GET()	string id	ServiceDescription	
POST()	TechnicalServiceDescription newDescription	boolean	requires HTTP Basic auth credentials

ServiceRegistrations/{service_id}/servicedescriptions/servicedatadescriptions resource for updating, retrieving and removing service data descriptions utilized in the Service Registry:

Operations

Method	Params [type, name, notes]	Return values	Notes
DELETE()	string id	boolean	requires HTTP Basic auth credentials
GET()	string id	ServiceDescription	
POST()	ServiceDataDescription newDescription	boolean	requires HTTP Basic auth credentials

Instantiation

After a Service has been registered and described, the last phase is linking service endpoints providing the particular interface to the description (Service instantiation). A service instance is an entity that implements the described service interfaces (technical description) and can be accessed through a URL. For each Technical Service Description, there can be multiple service instances.

ServiceRegistrations/{service_id}/serviceinstances resource for creating, retrieving and removing service instances implementing a particular technical service description in the Service Registry:

Operations

Method	Params [type, name, notes]	Return values	Notes
DELETE()	string id	boolean	requires HTTP Basic auth credentials
GET()	string id	ServiceDescription	
POST()	ServiceInstance newInstance	boolean	requires HTTP Basic auth credentials

2.2 Service Registry Data Model

This chapter describes the data model of the Service Registry and give details of the data model components.

Figure 2.3 shows the information stored on each service. The only “MUST HAVE” information on each Service Description is the name and a version number. For efficient service integration, a Technical Service Description (such as WADL for REST) and a semantic description of the data provided or consumed by the service are enforced. Each registered Service Description is given a Service ID that is unique in the context of this registry and can be used in linking services (Sources or Sinks) together.

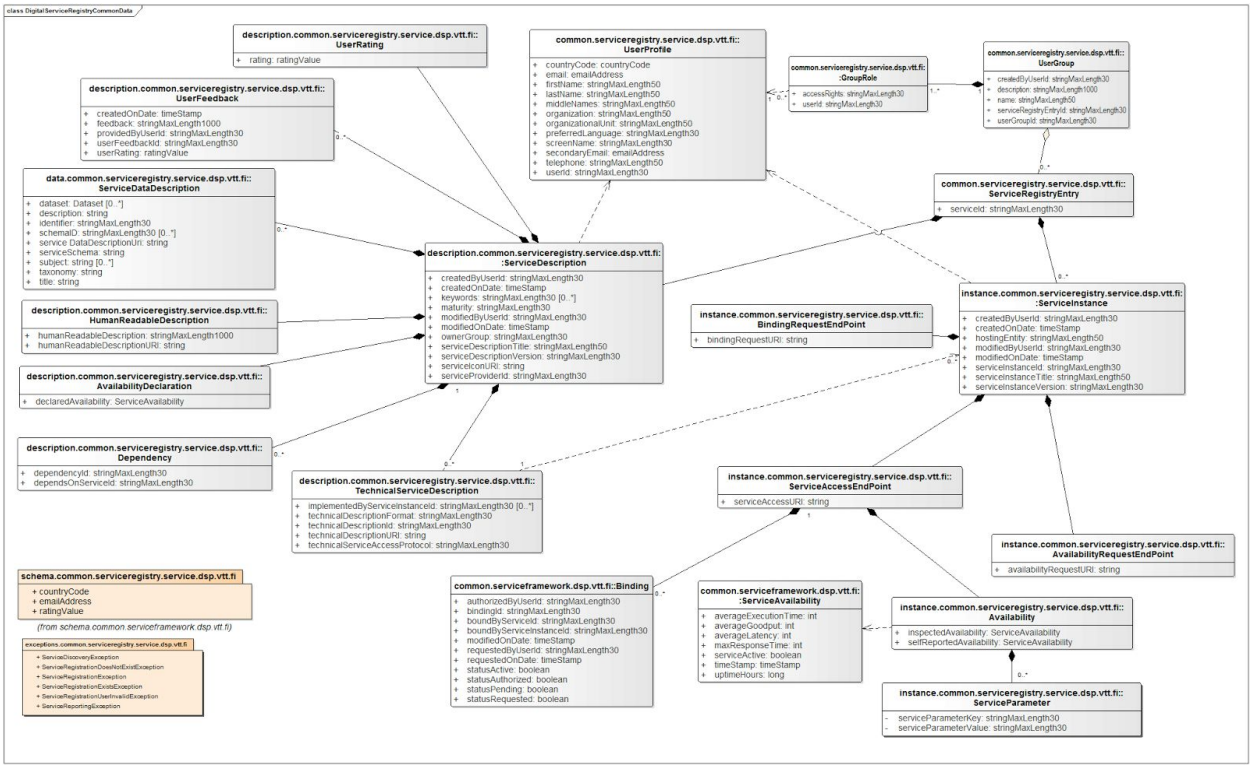

Figure 2.3. Service Registry data model

The schema definition can be found at:
https://github.com/digitalserviceshub/serviceregistry/blob/master/serviceregistry-common/src/main/resources/description.common.serviceregistry.service.dsp.vtt.fi.xsd

The focus here is to describe different parts of the Service Description. Details about other components in the Service Registry model can be found in the Appendix 2 - Common Service Registry data.

2.2.1 Service Description

ServiceDescription

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Main entity for the identified and registered service

Attributes

Attribute	Type	Constraints and tags	Notes
serviceDescriptionTitle	stringMaxLength50	*Default:*	Well descriptive title of the service description
serviceDescriptionVersion	stringMaxLength30	*Default:*	Version identifier of the service description following developer's own format
serviceIconURI	string	*Default:*	URI to the logo representing the service in the registry
maturity	stringMaxLength30	*Default:*	Maturity of the service description
keywords Collection [0..*]	stringMaxLength30	*Default:*	Selected set of keywords describing the service
serviceProviderId	stringMaxLength30	*Default:*	Identification of the service provider
createdOnDate	timeStamp	*Default:*	Date this service description was created

modifiedOnDate	timeStamp	*Default:*	Date this service description was modified last time
modifiedByUserId	stringMaxLength30	*Default:*	
createdByUserId	stringMaxLength30	*Default:*	

Human Readable Description

A Human Readable Description works as a base for the app/service store like operation on a MyData Operator.

Attribute	Type	Constraints and tags	Notes
humanReadableDescriptionURI	string	*Default:*	URI for the provided human readable description or documentation
humanReadableDescription	stringMaxLength1000	*Default:*	Provided human readable description or document content

Technical Service Description

A Technical Service Description contains a machine-readable API description of the service interface (e.g. WADL for REST-type service interface). This is essential for service developers for enabling service integration.

Attribute	Type	Constraints and tags	Notes
technicalDescriptionId	stringMaxLength30	*Default:*	Unique identifier of the technical service description
technicalDescriptionURI	string	*Default:*	URI to the service description
implementedByServiceInstanceId	stringMaxLength30	*Default:*	Service instance identifier of the service that implements this technical description
technicalServiceAccessProtocol	stringMaxLength30	*Default:*	Service access protocol identifier intended to be used with the service technically described

Service Data Description

A Service Data Description - metadata about data of a service - is provided as a Source or required as a Sink by the service. A Service Data Description for a Source describes what kind of data it produces (output data) and from where and in which format it is available. A Service Data Description for a Sink describes what kind of data it uses and needs (type of input data) and it does not include distribution information. The Service Data Description enables discovery of compatible services (Sink - Source) besides on technical interoperability level (Matching interface and data format), also on semantic interoperability level (e.g. discovery of all HR services regardless of data format) and implementation of data translators.

In order to be able to describe different features of data and to support versatile service discovery there are different type of requirements for service data description such as

- Ability to describe multiple data sets of a service
- Support for using semantic linking (existing taxonomies, Linked Data)
- General description of a dataset (title, description, keywords), supporting different languages
- Subject of data (grocery data, health of data..), preferable described with some defined taxonomy
- Information about a publisher of a data set
- Service data type – output (source) / input (sink)
- Data format (format, type, size) and accessibility (from where it is available)
- Update frequency and type (e.g. batch, real time, poll)
- Release and modification dates
- Rights and licences
- Information about time interval and spatial related data
- Devices from where data is collected
- Data structure definition, including available data components

There are existing vocabularies which define datasets and their structure and support many of these requirements. These vocabularies are developed for increasing findability of datasets and data items and interoperability between datasets:

- Data Catalog vocabulary – DCAT (http://www.w3.org/TR/vocab-dcat/) is design to facilitate interoperability between data catalogs published on the Web and to increase discoverability and enable applications easily to consume metadata from multiple catalogs. DCAT-vocabulary describes concepts such as data catalog, data set, distribution and a publisher.
- The RDF Data Cube Vocabulary –QB (http://www.w3.org/TR/vocab-data-cube/) is design to publish multi-dimensional data, such as statistics, on the web in such a way that it can be linked to related data sets and concepts. QB-vocabulary supports defining a data set and its data structure and components (metadata of data).

Both of these vocabularies are published as W3C Recommendation 2014 and they are used also for describing Open Data Datasets, which have same challenges of data findability and interoperability as different datasets of personal data.

Semantic Sensor Network Ontology (ssn) ontology (http://www.w3.org/2005/Incubator/ssn/ssnx/ssn) describes sensors and observations, and related concepts. It defines concepts such as sensor data, sensing device, feature of interest (e.g. person) and observation. It can be used for example describing measurement and activity data from different Fitness gadgets or a house related sensor data.

Definitions support semantic matching of concepts and linking definitions to Linked Data and common classification schemes. This supports creating common understanding of data and interoperability between different systems and services. Different classification schemes such as categories for personal data, service providers or health data can be used in definitions.

The classification schemes are represented using the SKOS vocabulary (http://www.w3.org/TR/2009/REC-skos-reference-20090818/). The SKOS data model provides a standard for porting different thesauri, taxonomies, classification schemes and subject heading to the Semantic web. A SKOS concept scheme can be viewed as an aggregation of one or more SKOS concepts that is used to describe categories, themes and subjects in datasets in the catalog. Concepts are organized into informal hierarchies and association networks.

A data model for a Service Data Description

The above ontologies are used in creating a data model for the Service Data Description. Specifications for both XML and RDF versions of the service data model have been created. RDF version of the data model supports full capability of expression of imported ontologies. The XML based service data description is a simplified version. First the overview of semantic version of the data model will be presented, and then the main classes and properties will be presented in more detailed.
The ServiceData ontology that describes the structure of Service Data Description can be seen at Appendix 3. The ServiceData ontology imports the following ontologies and the relations of the main classes are presented in Figure 2.4:

```
<owl:Ontology rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData">
        <owl:imports rdf:resource="http://purl.oclc.org/NET/ssnx/ssn"/>
        <owl:imports rdf:resource="http://purl.org/dc/terms/"/>
        <owl:imports rdf:resource="http://purl.org/linked-data/cube"/>
        <owl:imports rdf:resource="http://www.w3.org/ns/dcat"/>
</owl:Ontology>
```

The namespaces used in the descriptions are:

Prefix	**Namespace**
dcat	http://www.w3.org/ns/dcat#
dc	http://purl.org/dc/elements/1.1/

dct	http://purl.org/dc/terms/
qb	http://purl.org/linked-data/cube#
skos	http://www.w3.org/2004/02/skos/core#
ssn	http://purl.oclc.org/NET/ssnx/ssn#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
foaf	http://xmlns.com/foaf/0.1/

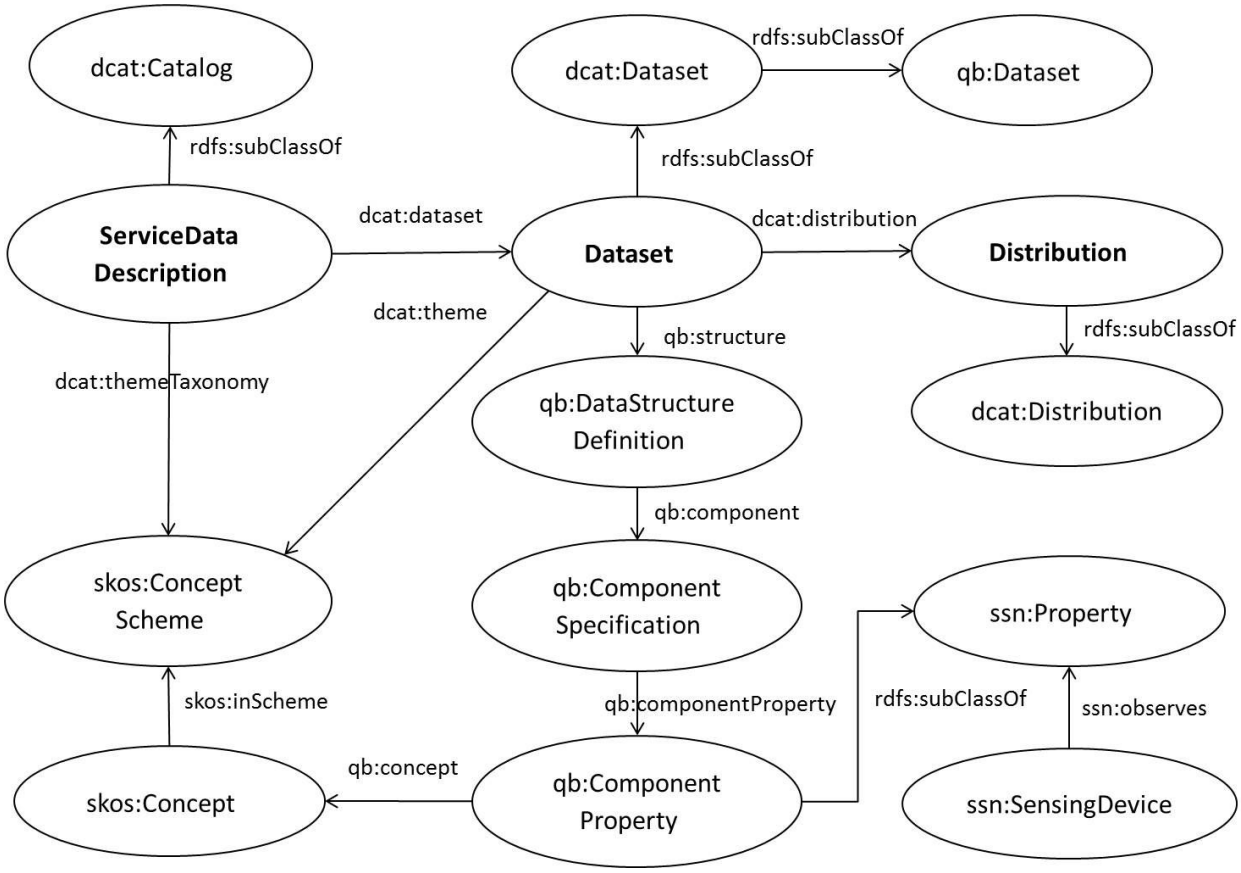

Figure 2.4. The main classes and their relationships in the ServiceData ontology. This supports general description of datasets and data elements of sources by importing and combining DCAT, QB and SSN ontologies. These ontologies builds upon existing RDF vocabularies such as SKOS, Duplin Core Terms and FOAF. Note that the figure do not present all the classes of the imported ontologies.

In figure 2.5 is presented the main components of the service data description. Service Data Description collects information about different datasets of a service. A service may have one or several datasets e.g. shopping data may be divided to different datasets from "shopping data as product level" to "shopping data as product category level" and to "a classification of life situation" or fitness API of a service may be divided to activity, measure and sleep datasets. Each instance of dataset description defines general metadata such as title, description, dates, publisher, distribution and structure of data in the dataset. A publisher refers to an organisation who is offering a service. Distribution includes information about from where data is available, in which format and under which conditions and it is only defined for a source. When describing a service, a service provider may have some license conditions. A Data structure and its components and properties are defined with DataStructureSpecification, ComponentSpecification and ComponentProperty classes. For a Source the structure describes what kind of data it produces and for a Sink what kind of data a Sink uses or needs.

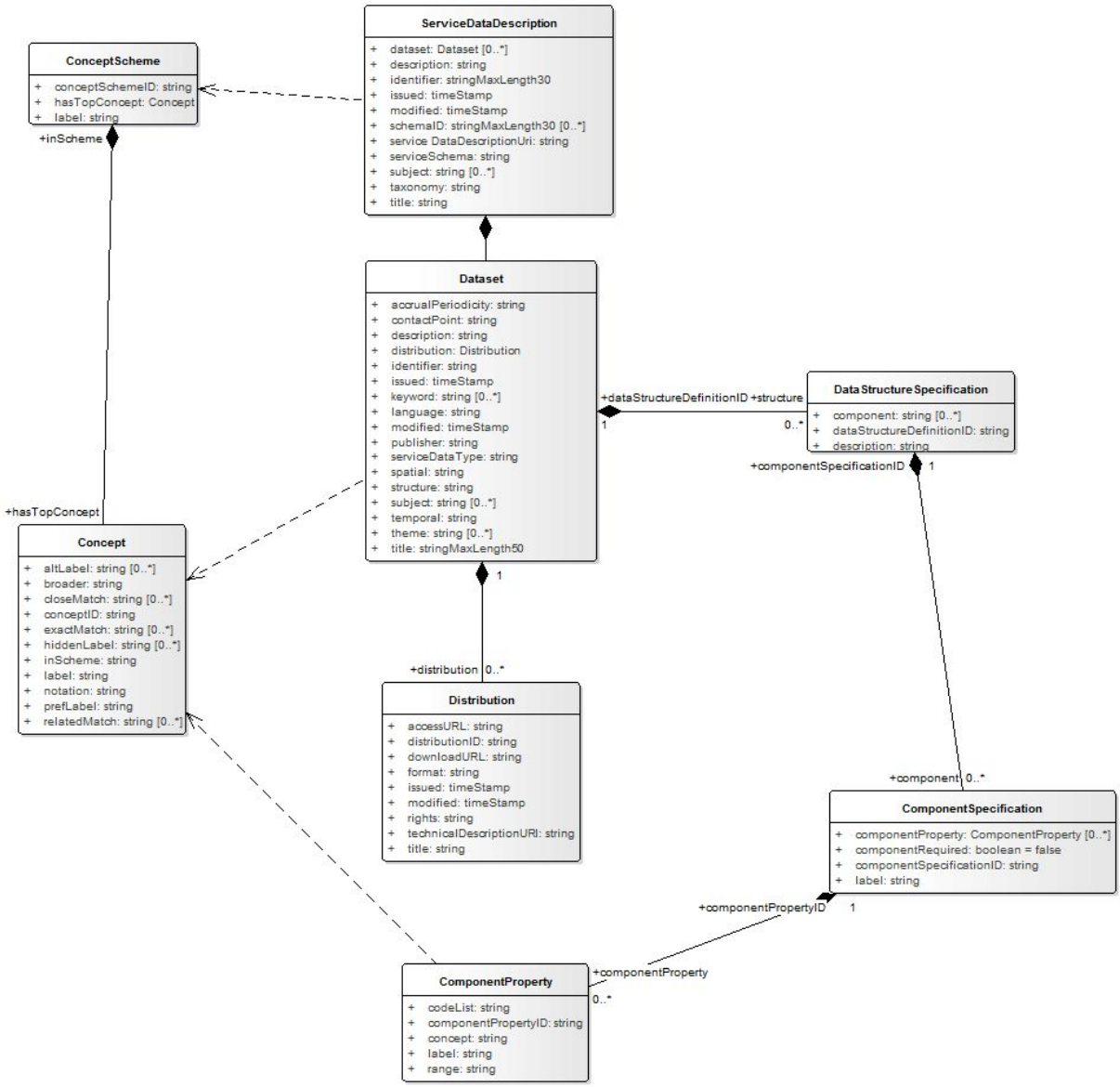

Figure 2.5 The components of the Service Data Description

The following tables describe classes and their attributes. Relations of attributes to supporting vocabulary are described in parenthesis. In addition the data model specification includes Service Registry and MyData service specific attributes, which are highlighted with italic font.

Service Data Description (dcat:Catalog)

Collection of metadata about data of a service.

Attribute	**Type**	**Constraints**	**Notes**
identifier	string		A unique identifier of the service data description.
schemaID	string		Link (URI) to a schema used by a service registry to create a unified service data description of the service.
description (dct:description)	string		Free text description of service data description
title (dct:title)	string		A name given to a service data description
subject [0..*] (dct:subject)	string		Links (URI) to existing taxonomies, vocabularies and Linked Data resources describing the service
serviceDataDescriptionURI	string		Link to a document that describes the service data description based on the defined schema (schemaID) (=serviceDataDescriptionDocument)
taxonomy (dcat:themeTaxonomy, qb:codeList))	string		Link to taxonomy that is used to describe data of a service (dcat:themeTaxonomy & qb:codeList)
serviceSchema	string		Link to the original schema of the service.
dataset [0..*}] (dcat:dataset)	Dataset		Reference to a data set that is part of service data. (dcat:dataset)
issued (dct:issued)	timeStamp		Date when this Service data description was created.
modified (dct:modified)	timeStamp		Date when this Service data description was modified.

Dataset (dcat:Dataset)

A Dataset is a collection of data that corresponds to a defined structure.

Attribute	Type	Constraints	Notes
identifier	string		A unique identifier of the dataset.
subject [0..*] (dct:subject)	string		Links (URI) to existing taxonomies, vocabularies and Linked Data resources describing the service
serviceDataType	string	value: 'input', 'output' or 'internal'	'input' describes data a Sink can receive from an external source 'output' describes data a Source can share 'internal' describes data a service uses internally
title (dct:title)	stringMaxLength 50		A name given to the dataset. In multiple languages.
description (dct:description)	stringMaxLength 50		Free text description of the dataset. Possibly in multiple languages
theme [0..*] (dcat:theme)	string		The main category(ies) of the dataset, expressed as a semantic URI
issued (dct:issued)	timeStamp		Date when this dataset description was created
modified (dct:modified)	timeStamp		Date when this dataset description was modified
language (dct:language)	string		The natural language of the dataset.
publisher (dct:publisher)	string		The entity responsible for making the dataset online.
spatial (dct:spatial)	string		Spatial coverage of the dataset.
temporal (dct:temporal)	string		The temporal period that the dataset covers.
contactPoint (dcat:contactPoint)	string		Relevant contact information of a dataset
keyword [0..*] (dcat:keyword)	string		A keyword describing a dataset. Possible in multiple languages.
accrualPeriodicity (dct:accrualPeriodicity)	string		The frequency at which dataset is published.
structure [0..*] (qb:structure)	DataStructureDefinition		Reference to a data structure definition that defines the dataset structure.

distribution [0..*] (dcat:distribution)	Distribution		Reference to a specified distribution of the dataset.
purpose [0..]*	String		Defines the purpose for which a service uses the data.

Distribution (dcat:Distribution)

Represents a specific available form of a dataset. Each dataset might be available in different forms, these forms might represent different formats of the dataset or different endpoints. Examples of distributions include a downloadable CSV file, an API or an RSS feed.

Attribute	**Type**	**Constraints**	**Notes**
distributionID	string		A unique identifier of the distribution.
title (dct:title)	string		A name given to the distribution.
accessURL (dcat:accessURL)	string		A landing page, feed, API endpoint or other type of resource that gives access to the distribution of the dataset
downloadURL (dcat:downloadURL)	string		A file that contains the distribution of the dataset in a given format
technicalDescriptionURI	string		Link (URI) to the technical description document (defined in the technical service description).
format (dct:format)	string		The file format of the distribution, e.g. application/vnd.ms-excel, application/json
rights (dct:rights)	string		Link to a licence / rights document. e.g. link to a Service's Terms of Service - document
issued (dct:issued)	timeStamp		Date when this description was created
modified (dct:modified)	timeStamp		Date when this description was modified

DataStructureDefinition (qb:DataStructureDefinition)

Defines the structure of one or more datasets.

Attribute	**Type**	**Constraints**	**Notes**
dataStructureDefinitionID	string		A unique identifier of the data structure

			definition.
description (dct:description)	string		Free text description of the data structure definition. Possibly in multiple languages
component [0..*] (qb:component)	ComponentSpecification		Reference to a specified component specification that is part of the data structure definition.

ComponentSpecification (qb:ComponentSpecification)

Used to define components.

Attribute	Type	Constraints	Notes
componentSpecificationID	string		A unique identifier of the component specification
label (rdfs:label)	string		A name of a component
componentProperty [0..*] (qb:componentProperty)	ComponentProperty		Reference to a specified component property that is part of the component specification “hascomponentProperty”
componentRequired (qb:componentRequired)	boolean		Indicates whether a component is required (true) or optional (false) by a Sink. Default is false (optional)

ComponentProperty (qb:ComponentProperty)

Definition of component properties.

Attribute	Type	Constraints	Notes
componentPropertyID	string		A unique identifier of the component property
label (rdfs:label)	string		A name of a component property
range (rdfs:range)	string		type of the value of the property (e.g. xsd:int…)
codeList (qb:codeList)	string		Link (URI) to a definition of the code list associated with a component property
concept	string		Links (URI) to existing taxonomies, vocabularies describing

(qb:concept)			the property

The SKOS vocabulary is used in defining the attributes for ConceptScheme and Concept. These are used to describe taxonomies, categories, themes, subjects and code lists in the dataset. Definitions support properties for semantic enrichment and linking to be used in semantic service discovery.

ConceptScheme (skos:ConceptScheme)

Defines a set of concepts. Is used to define concept schemes used by the dataset.

Attribute	**Type**	**Constraints**	**Notes**
conceptSchemeId	string		A unique identifier of the concept scheme
label (rdfs:label)	string		A name given to the concept schema.
hasTopConcept (skos:hasTopConcept)	Concept		A top level concept in the concept scheme.

Concept (skos:Concept)

Defines a concept, its labels and relations to other concepts. Defines concepts used by the dataset.

Attribute	**Type**	**Constraints**	**Notes**
conceptID	string		A unique identifier of the concept
label (rdfs:label)	string		A name given to the concept.
prefLabel (skos:prefLabel)	string		The preferred lexical label for a resource, in a given language.
altLabel (skos:altLabel)	string		An alternative lexical label (e.g. acronyms, different labels of the same concept) for a resource.
hiddenLabel (skos:hiddenLabel)	string		A lexical label (e.g. misspellings, singular/plural) for a resource, accessible to free text search operations.
inScheme (skos:inScheme)	string		Relates a concept to the concept scheme that it is a top level concept of.
notation (skos:notation)	string		A classification code
broader (skos:broader)	string		Links a concept that is more general in meaning.

closeMatch (skos:closeMatch)	string		Links two concepts that are sufficiently similar. Used in semantic mappings between concepts in different schemas.
exactMatch (skos:exactMatch)	string		Links two concepts that are similar. Used in semantic mappings between concepts in different schemas.
relatedMatch (skos:relatedMatch)	string		Links two concepts that are related. Used in semantic mappings between concepts in different schemas.

Creation of a Service Data Description

Service Data Description is composed from different parts supporting service discovery, data mapping, matching and integration at different levels. The Service Data Description is OPTIONAL, but the level of the Service Data Description of Source or Sink defines how intelligent and automatic service discovery and data integration can be made. The Service Registry needs to provide methods and tools to support creation of the Service Data Description. The table below describes creation of service data description in different levels and phases and defines a purpose of them. In order to be able to support these functionalities at defined level the specific part of the Service Data Description SHOULD be made. Different phases of data description input will create machine readable Service Data Schema document, which link is added to serviceDataDescriptionURI property in ServiceDataDescription class.

Class	**Description / Phase**	**Purpose**	**Needed support for creation of data**
Service Data Description (dcat:Catalog)	Creating basic description of the service data (e.g. title, description) Describes what kinds of datasets are available by a service.	service discovery based on general descriptions (free text searching) human readable descriptions of data	Web forms with input fields and transformation of given data into the schema of the service registry.
Dataset (dcat:Dataset)	Creating basic description of a specific dataset e.g. title, description, keywords Describes in general level what kind of dataset is available	service discovery (free text searching) human readable description of dataset	Web forms with input fields and transformation of given data into the schema of the service registry

Distribution (dcat:Distribution)	Access information, dates, format, rights, link to a technical description document This is created only for a source.	data accessibility information service integration	Web forms with input fields and transformation of given data into the schema of the service registry
Structure of dataset; DataStructureSpecification, ComponentSpecification, ComponentProperty (qb:DataStructureSpecification, qb:ComponetSpecification, qb:ComponentProperty)	Describing the structure of the dataset including its components and properties.	Creation of machine readable description of components and properties of datasets service discovery at component and property level support for automatic integration human readable descriptions of data components of a dataset	The service may have its own schema or it may be csv or JSON description. Functionality is needed to read, map and transform the service data based on the common schema of the service registry.
Semantic enrichment of data descriptions will add information to different properties in different classes; Service data description; subject and taxonomy properties. Dataset; subject and theme properties, ComponentProperty; codeList and concept properties. ConceptScheme and Concept classes and their properties are used for modelling taxonomies and vocabularies.	Semantic enrichment of labels, keywords, properties and concepts	intelligent service discovery and automatic mapping and matching of data Adds support for multilingualism, finding same concepts with different labels, utilising relations of concepts in searching and matching	Methods and tools for semantic enrichment; adding links to common vocabularies and Linked Data, utilising this data to enrich information of different concepts. (e.g. multilingual labels, relations of concepts) Semantic modelling and enrichment of existing taxonomies, categories or coded lists of a service. Modelling the result based on the common schema of the service registry.

An example of Service Data Description

Below is presented some examples of Service Data Description in RDF format. W2E (https://w2e.fi/#/) - Source is used as an example. W2E service offers several APIs (https://w2e.fi/doc/introduction.html), meaning that each API description will have its own dataset description. In this example is used only part of the Category data - Get measure API description (https://w2e.fi/doc/data_category.html). The purpose is to give an example of usage of the Service Data Description data model, not to make complete data description of W2E service.

An example of basic information of Service Data Description as well as links to different datasets of the service:

```
<rdf:Description rdf:about="#Catalog_W2E">
        <rdf:type rdf:resource="http://www.w3.org/ns/dcat#Catalog"/>
        <rdf:type rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#ServiceDataDescription"/>
        <dct:description xml:lang="en">Wellness Warehouse Engine (W2E) is a tool for collecting
        user's wellness related data. Data can be related to activity, measurements or sleep and it can
        be requested as raw, unify and category data. For each type is defined its own API description.
        See documentation at https://w2e.fi/doc/introduction.html.</dct:description>
        <dct:title xml:lang="en">Service data description of Wellness Warehouse Engine</dct:title>
        <dct:issued>2015-12-15T11:38:00</dct:issued>
        <dct:modified>2015-12-16T09:30:00</dct:modified>
        <foaf:homepage rdf:resource="https://w2e.fi/#/"/>
        <dcat:dataset rdf:resource="#Dataset_W2E_Category_Measure"/>
        <dcat:dataset rdf:resource="#Dataset_W2E_Category_Activity"/>
        <dcat:dataset rdf:resource="#Dataset_W2E_Category_Sleep"/>
        <dcat:dataset rdf:resource="http://w2e.fi/doc/dhr/Dataset_W2E_Category_Measure"/>
        <sdata:schemaID rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData"/>
        <sdata:serviceDataDescriptionURI rdf:resource="https://w2e.fi/doc/dhr/w2e.rdf"/>
        <dc:identifier>w2e1000</dc:identifier>
</rdf:Description>
```

An example of a dataset description:

```
<rdf:Description rdf:about="#Dataset_W2E_Category_Measure">
        <rdf:type rdf:resource="http://www.w3.org/ns/dcat#Dataset"/>
        <dcat:distribution rdf:resource="#Distribution_W2E_Category_Measure"/>
        <qb:structure rdf:resource="#DataStructureDefinition_W2E_Category_Measure_1"/>
        <rdf:type rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData##Dataset"/>
        <dct:description xml:lang="en">Data is grouped from all available sources into categories.
        This describes measure category. See documentation at http://w2e.fi/doc/data_category.html.
        </dct:description>
        <dct:title xml:lang="en">Category data - get measure</dct:title>
        <dcat:keyword xml:lang="en">weight</dcat:keyword>
        <dcat:keyword xml:lang="en">blood pressure</dcat:keyword>
        <dcat:keyword xml:lang="en">heart rate</dcat:keyword>
        <dcat:keyword xml:lang="en">Fitbit</dcat:keyword>
        <dcat:keyword xml:lang="en">Runkeeper</dcat:keyword>
        <dcat:keyword xml:lang="en">Withings</dcat:keyword>
        <sdata:serviceDataType rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >output</sdata:serviceDataType>
 </rdf:Description>
```

An example of a distribution description:

```
<rdf:Description rdf:about="#Distribution_W2E_Category_Measure">

        <rdf:type rdf:resource="http://www.w3.org/ns/dcat#Distribution"/>
        <foaf:homepage rdf:resource="https://w2e.fi/doc/data_category.html"/>
        <dcat:accessURL rdf:resource="https://w2e.fi/doc/data_category.html"/>
        <dct:format>application/json</dct:format>
 </rdf:Description>
```

An example of a structure definition, which includes definitions of DataStructureDefinition, ComponentSpecification and ComponentProperty.

An example of DataStructureDefinition:

```
<rdf:Description rdf:about="#DataStructureDefinition_W2E_Category_Measure_1">
        <rdf:type rdf:resource="http://purl.org/linked-data/cube#DataStructureDefinition"/>
        <qb:component rdf:resource="#ComponentSpecification_W2E_Category_Measure_heartRate"/>
        <qb:component rdf:resource="#ComponentSpecification_W2E_Category_Measure_weight"/>
 </rdf:Description>
```

Examples of component specification descriptions:

```
<rdf:Description rdf:about="#ComponentSpecification_W2E_Category_Measure_heartRate">
        <rdf:type rdf:resource="http://purl.org/linked-data/cube#ComponentSpecification"/>
        <qb:componentProperty rdf:resource="#heartRate"/>
        <qb:componentProperty rdf:resource="#date"/>
        <qb:componentProperty rdf:resource="#source"/>
        <qb:componentProperty rdf:resource="#timezone"/>
        <rdfs:label xml:lang="en">heartRate</rdfs:label>
</rdf:Description>

<rdf:Description rdf:about="#ComponentSpecification_W2E_Category_Measure_weight">
        <rdf:type rdf:resource="http://purl.org/linked-data/cube#ComponentSpecification"/>
        <qb:componentProperty rdf:resource="#weight"/>
        <rdfs:label xml:lang="en">weight</rdfs:label>
        <qb:componentProperty rdf:resource="#date"/>
        <qb:componentProperty rdf:resource="#fatFree"/>
        <qb:componentProperty rdf:resource="#fatMass"/>
        <qb:componentProperty rdf:resource="#source"/>
        <qb:componentProperty rdf:resource="#timezone"/>
</rdf:Description>
```

Examples of descriptions of component properties

```
<rdf:Description rdf:about="#heartRate">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">heart rate {@en}</rdfs:label>
        <skos:altLabel xml:lang="en">pulse</skos:altLabel>
        <skos:altLabel xml:lang="en">pulse rate</skos:altLabel>
        <qb:concept rdf:resource="http://www.yso.fi/onto/mesh/D006339"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
        <rdf:type rdf:resource="http://purl.org/linked-data/cube#ComponentProperty"/>
</rdf:Description>

<rdf:Description rdf:about="#date">
        <rdfs:label xml:lang="en">date</rdfs:label>
        <rdfs:label xml:lang="fi">päivämäärä</rdfs:label>
        <qb:concept rdf:resource="http://purl.org/dc/terms/date"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
        <rdf:type rdf:resource="http://purl.org/linked-data/cube#ComponentProperty"/>
</rdf:Description>
```

An example of semantically enriched information:
The keywords of the dataset description are enriched with links to other vocabularies and added to the dataset description. This information can be used to automatically add multilingual keywords and alternative terms (for example see information attached to heart rate at

http://www.yso.fi/onto/mesh/D006339) to the dataset description in order to improve findability.

```
<rdf:Description rdf:about="#Dataset_W2E_Category_Measure">
        <dct:subject rdf:resource="http://www.yso.fi/onto/mesh/D001835"/>
        <dct:subject rdf:resource="http://www.yso.fi/onto/mesh/D001794"/>
        <dct:subject rdf:resource="http://www.yso.fi/onto/mesh/D006339"/>
        <dct:subject rdf:resource="http://dbpedia.org/resource/Fitbit"/>
        <dct:subject rdf:resource="http://dbpedia.org/resource/RunKeeper"/>
        <dct:subject rdf:resource="http://dbpedia.org/resource/Withings"/>
</rdf:Description>
```

A JSON-LD example

Service Data Description can be presented also as JSON-LD (JavaScript Object Notation for Linked Data) format. Below is a short example of definitions of a data structure in JSON-LD format:

```
{
"@context": {
        "w2e": "http://w2e.fi/doc/dhr#",
        "xsd": "http://www.w3.org/2001/XMLSchema#",
        "qb": "http://purl.org/linked-data/cube#",
        "rdfs": "http://www.w3.org/2000/01/rdf-schema#",

        "range":
        {
        "@id": "http://www.w3.org/2000/01/rdf-schema#range",
        "@type": "@id"
        },
        "concept":
        {
        "@id": "http://purl.org/linked-data/cube#concept",
        "@type": "@id"
        },
        "componentProperty":
        {
        "@id": "http://purl.org/linked-data/cube#componentProperty",
        "@type": "@id"
        },
        "component":
        {
        "@id": "http://purl.org/linked-data/cube#component",
        "@type": "@id"
        }
        },

"@id": "w2e:DataStructureDefinition_W2E_Category_Measure_1",
"@type": "qb:DataStructureDefinition",
```

```
"component":[
   {
   "@id": "w2e:ComponentSpecification_W2E_Category_Measure_heartRate",
   "@type": "qb:ComponentSpecification",
   "componentProperty": [
            {
            "@id": "w2e:heartRate",
            "@type": "qb:ComponentProperty",
            "rdfs:label": "heart rate",
            "range": "xsd:int",
            "concept": "http://www.yso.fi/onto/mesh/D006339"
            },
            {
            "@id": "w2e:date",
            "@type": "qb:ComponentProperty",
            "rdfs:label": "date",
            "range": "xsd:date",
            "concept": "http://purl.org/dc/terms/date"
            }
            ]
   }
 ]
}
```

3. Appendix #1 Service Registry ROA Interface

This appendix illustrates ROA style interface with abstract CRUD (Create, Read, Update, Delete) operations of the Service Registry. The identification and addressing of the ROA resources in this Platform Independent Model is based on the usage of URIs as identifications. For the sake of clarity in illustrations, they have been divided to two parts; the Figure 1 for registering service descriptions and the Figure 2 for service instance registration purposes. These figures illustrate the resources and structural relations between them with URI based addressing. The specified resources have abstract CRUD operation stereotypes with the actual suggested method. The URI templates and the mapping of the URI and the abstract CRUD stereotype operations are defined in this appendix along with the suggested resource methods. The resource construction and used data objects are based on the data model presented in Appendix #2 Common service registry data. The access control is considered here as an external aspect and thus an implementation detail. Refer to the chapter 3.1.4 for details.

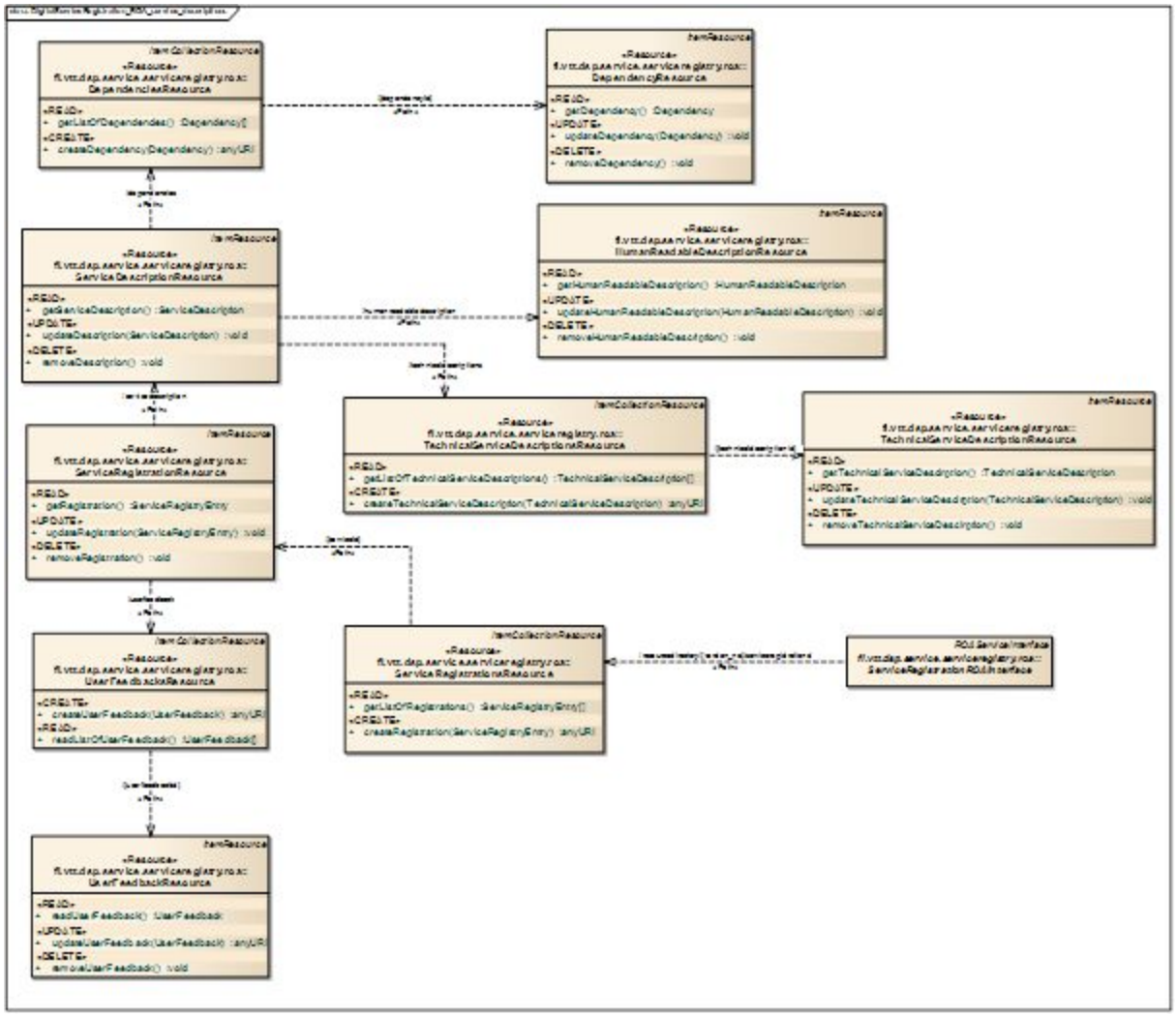

Figure 1. Service registry ROA style service interface for registering service descriptions

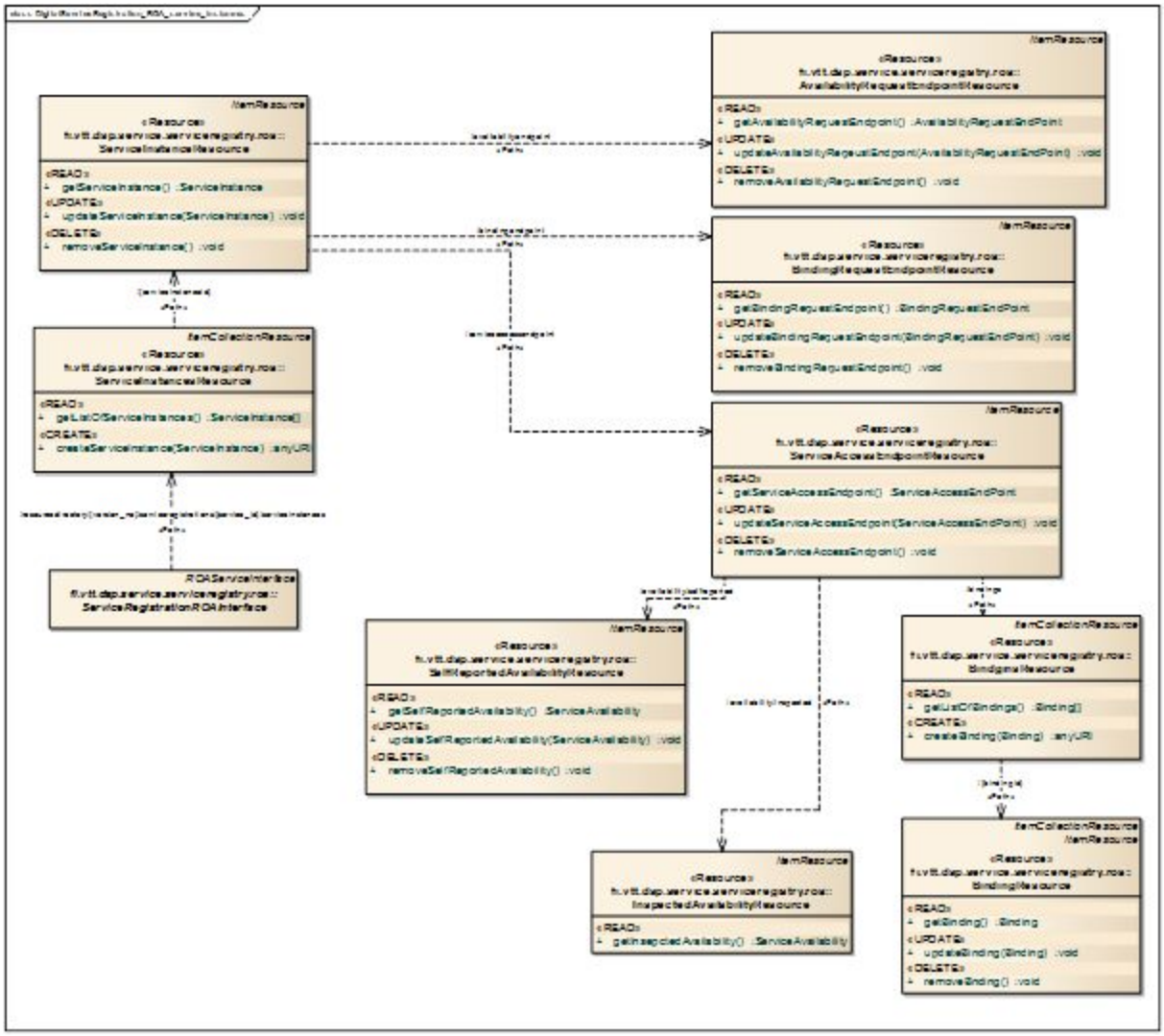

Figure 2. Service registry ROA style service interface for registering service instances.

The following chapters gives technical details of the specified service interface starting with the URIs, CRUD and URI mapping and the detailed resource specifications.

3.1 Service registry base URI

For the ROA style service interface base URI for the service registry access is

/resourcedirectory/{version_no}/serviceregistrations/

in where, version numbering currently will be ‘v1’.

3.2 Service Registry URI templates

The following additional URI templates can be attached to each of the documented resources for service discovery and pagination purposes.

3.2.1 Service discovery

The ROA style interface supports FIQL (http://cxf.apache.org/docs/jax-rs-search.html) searches. Thus, registered services, service descriptions and instances can be searched and discovered at their respective URIs by using the following URI template
{resource_uri}?_s={parametername1}=={parametervalue1}

where parametername is the name of the parameter used to query matching resources and parametervalue the corresponding value. Refer to the FIQL page for more complex query methods.

3.2.2 Pagination

The results of queries or other list retrievals can be paginated on the client by using following result limitters

{resource_uri}/?top=N

returning N results from the top of the result set

{resource_uri}/?tail=N

returning N results starting from the beginning of the result set

{resource_uri}/?section=N-M

returning a section of the results between N and M results from the beginning of the result set. The total number of items in collection type resources can be access as described in the ROA framework specification.

3.3 Service description registration

The following table represents a mapping of the URI and abstract CRUD operations with their semantics that have been defined for each of the resources for service description registration purposes. To successfully add a service description, the service itself must be registered first with ServiceRegistrations resource.

Resource name	URI	READ	UPDATE	CREATE	DELETE
ServiceRegistrations	/	Returns a list of the current available registrations as ServiceRegistration[]	N/A	Creates a new service registration with	N/A

				input ServiceRegistration to a returned URI with service id	
ServiceRegistration	/{service_id}	Returns a representation of the identified service registry entry as ServiceRegistration	Updates the identified service registration with input ServiceRegistration	N/A	Removes the identified service registration
ServiceDescription	/{service_id}/servicedescription	Returns a service description representation of the identified service as ServiceDescription	Updates service description of the identified service with input ServiceDescription	N/A	Removes identified service's service description
HumanReadableDescription	/{service_id}/servicedescription/humanreadabledescription	Returns a human readable service description of the identified services as HumanReadableDescription	Updates the identified service's human readable service description with input HumanReadableDescription	N/A	Removes the identified service's human readable service description
TechnicalServiceDescription	/{service_id}/servicedescription/technicalescriptions	Returns a list of the technical service description of the identified services as TechnicalServiceDescription[]	Updates the identified service's technical description	N/A	Removes the identified service's technical description

			with input TechnicalServiceDescription		
Dependencies	/{service_id}/servicedescription/dependencies	Returns a list of the current dependencies of the identified services as Dependency[]	N/A	Adds a new dependency with input Dependency and returns a URI with dependency id	N/A
Dependency	/{service_id}/servicedescription/dependencies/{dependency_id}	Returns a representation of the identified dependency as Dependency	Updates the identified dependency with input Dependency	N/A	Removes the identified dependnecy
UserFeedbacks	/{service_id}/servicedescription/userfeedbacks	Returns a list of the user provided feedback for the service description as UserFeedback[]	N/A	Add a new user provided feedback with input UserFeedback for the service description	N/A
UserFeedback	/{service_id}/servicedescription/userfeedbacks/{userfeedback_id}	Returns a representation of the user feedback identified as UserFeedback	N/A	N/A	N/A

The input and output objects of

ServiceRegistration
HumanReadableDescription
TechnicalServiceDescription

Dependency
UserFeedback

refer to the data objects presented in the Appendix #1 Service Registry ROA Interface.

3.4 Service instance registration

The following table represents a mapping of the URI and abstract CRUD operations with their semantics that have been defined for each of the resources for service instance registration purposes. To successfully register a service instance, service itself and its service description must be registered first according to the Service description registration section above.

Resource name	URI	READ	UPDATE	CREATE	DELETE
ServiceInstancesResource	/{service_id}/serviceinstances	Returns a list of the given service's registered instances as ServiceInstance[]	N/A	Creates a new service instance with input ServiceInstance to a returned URI with instance id	N/A
ServiceInstanceResource	/{service_id}/serviceinstances/{serviceinstance_id}	Returns the given service instance of the given service as ServiceInstance	Updates the given service instance with input ServiceInstance on the given service	N/A	Removes the given service instance on the given service
AvailabiltyRequestEndpointResource	/{service_id}/serviceinstances/{serviceinstance_id}/availabilityendpoint	Returns the given service instance's availability request endpoint as AvailabiltyRequestEndpoint	Updates the given service instance's availability request endpoint with input AvailabiltyRequestEnd	N/A	Removes the given service instance's availability request endpoi

			point		nt
BindginRequestEndpointResource	/{service_id}/serviceinstances/{serviceinstance_id}/bindingendpoint	Returns the given service instance's binding request endpoint as BindginRequestEndpoint	Updates the given service instance's binding request endpoint with input BindginRequestEndpoint	N/A	Removes the given service instance's binding request endpoint
ServiceAccessEndpointResource	/{service_id}/serviceinstances/{serviceinstance_id}/serviceaccessendpoint	Returns the given service instance's service access endpoint as ServiceAccessEndpoint	Updates the given service instance's service access endpoint with input ServiceAccessEndpoint	N/A	Removes the given service instance's service access endpoint
SelfReportedAvailabilityResource	/{service_id}/serviceinstances/{serviceinstance_id}/serviceaccessendpoint/availability/selfreported	Returns a services availability information as reported by the service itself as ServiceAvailability	Updates the identified service instances self reported availability information with input ServiceAvailability	N/A	Removes the self reported availability information
InspectedAvailabilityResource	/{service_id}/serviceinstances/{serviceinstance_id}/serviceaccessendpoint/availability/inspected	Returns a services availability information as inspected by the registry as ServiceAvailability	N/A	N/A	N/A
BindingsResource	/{service_id}/serviceinstances/{serviceinstance_id}/serviceaccessendpoint/bindings	Returns a list of the registered bindgins on the service instance as	N/A	Creates a new binding to	N/A

		Binding[]		a service instance with input Binding	
BindingResource	/{service_id}/serviceinstances/{serviceinstance_id}/serviceaccessendpoint/bindings/{binding_id}	Returns the identified binding of the given service as Binding	Updates the identified binding with input Binding of the given service instance	N/A	Removes the identified binding on the given service instance

The input and output objects of

ServiceInstance
AvailabiltyRequestEndpoint
BindginRequestEndpoint
ServiceAccessEndpoint
ServiceAvailability
Binding
refer to the data objects presented in the .

3.5 Exceptions thrown

The exceptions thrown are contained in the common service registry data specification.

3.6 AvailabilityRequestEndpointResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Availability request endpoint resource for the registered service instance

Operations

Method	Params [type, name, notes]	Return values	Notes
getAvailabilityRequestEndpoint()		AvailabilityRequestEndPoint	Returns availability request endpoint resource representation

			on the given service instance
updateAvailabilityReqeustEndpoint()	AvailabilityRequestEndPoint updatedAvailabilityRequestEndpoint Updated availability request endpoint information	void	Updates availability request endpoint resource on the given service instance
removeAvailabilityRequestEndpoint()		void	Removes availability request endpoint resource from the given service instance

3.7 BindginsResource

Stereotype: «Resource»
Type: Class
Extends: ItemCollectionResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Collection resource for service instance's binding information

Operations

Method	Params [type, name, notes]	Return values	Notes
getListOfBindings()		Binding	Returns a list of the service instance's bindings
createBinding()	Binding createdBinding Binding information to be reported	anyURI	Creates a new binding item to the binding collection. Return URI to the created binding resource.

3.8 BindingRequestEndpointResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Binding request endpoint resource for the registered service instance

Operations

Method	Params [type, name, notes]	Return values	Notes
getBindingRequestEndpoint()		BindingRequestEndPoint	Returns binding request endpoint resource representation on the given service instance
updateBindingRequestEndpoint()	BindingRequestEndPoint updatedBindingRequestEndpoint Updated binding request endpoint information	void	Updates binding request endpoint resource on the given service instance
removeBindingRequestEndpoint()		void	Removes binding request endpoint resource from the given service instance

3.9 BindingResource

Stereotype: «Resource»
Type: Class
Extends: ItemCollectionResource, ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Resource for the individual binding information

Operations

Method	Params [type, name, notes]	Return values	Notes
getBinding()		Binding	Returns individual binding resource's representation associated with the registered service instance
updateBinding()	Binding updatedBinding Updated binding information	void	Updates the identified binding on the registered service instance
removeBnding()		void	Removes the identified binding from the registered service instance

3.10 DependenciesResource

Stereotype: «Resource»
Type: Class
Extends: ItemCollectionResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Main container resource for the dependencies on the service description

Operations

Method	Params [type, name, notes]	Return values	Notes
getListOfDependencies()		Dependency	Returns a list of the dependencies associated with the service description
createDependency()	Dependency dependency Dependency requested to be created	anyURI	Creates a new dependency for the service description. Return URI to the created dependency.

3.11 DependencyResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Resource for the individual dependency on the service description

Operations

Method	Params [type, name, notes]	Return values	Notes
getDependency()		Dependency	Returns a representation of the identified dependency on the service description
updateDependency()	Dependency updatedDependency	void	Updates the identified dependency in the service description

	Updated dependency information		
removeDependency()		void	Removes the identified dependency from the service description

3.12 HumanReadableDescriptionResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Resource for maintaining the human readable description in the identified service

Operations

Method	Params [type, name, notes]	Return values	Notes
getHumanReadableDescription()		HumanReadableDescription	Returns the human readable description for the identified service description
updateHumanReadableDescription()	HumanReadableDescription updatedHumanReadableDescription Updated human readable description	void	Updates the human readable description in the identified service description
removeHumanReadableDescription()		void	Removes the human readable description from the identified service description.

3.13 InspectedAvailabilityResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Operations

Method	Params [type, name, notes]	Return values	Notes
getInsepctedAvailability()		ServiceAvailability	Returns the service's inspected service availability information

3.14 SelfReportedAvailabilityResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Availability item resource holding availability information of the given service instance

Operations

Method	Params [type, name, notes]	Return values	Notes
getSelfReportedAvailability()		ServiceAvailability	Returns the service's reported service availability information
updateSelfReportedAvailability()	ServiceAvailability updatedAvailability Updated availability information	void	Updates the service's reported service availability information
removeSelfReportedAvailability()		void	Removes the service's reported service availability information

3.15 ServiceAccessEndpointResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Service access endpoint resource for the registered service instance

Operations

Method	Params [type, name, notes]	Return values	Notes
getServiceAccessEndpoint()		ServiceAccess EndPoint	Returns service access endpoint resource representation on the given service instance
updateServiceAccessEndpoint()	ServiceAccessEndPoint updatedServiceAccessPoint Updated service access endpoint information	void	Updates service access endpoint resource on the given service instance
removeServiceAccessEndpoint()		void	Removes service access endpoint resource from the given service instance

3.16 ServiceDescriptionResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Main resource of the individual service description

Operations

Method	Params [type, name, notes]	Return values	Notes

getServiceDescription()		ServiceDescription	Returns the identified service's description
updateDescription()	ServiceDescription updatedDescription Updated service description	void	Updates the identified service's description in the service registry
removeDescription()		void	Removes the identifies service's description from the service registry

3.17 ServiceInstanceResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Individual service instance item resource

Operations

Method	Params [type, name, notes]	Return values	Notes
getServiceInstance()		ServiceInstance	Returns representation of the identified service instance on the service registry
updateServiceInstance()	ServiceInstance updatedServiceInstance Updated service instance description information	void	Updates the given service instance description on the service registry
removeServiceInstance()		void	Removes identified service instance from the service registry

3.18 ServiceInstancesResource

Stereotype: «Resource»
Type: Class
Extends: ItemCollectionResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Service instance collection resource

Operations

Method	Params [type, name, notes]	Return values	Notes
getListOfServiceInstances()		ServiceInstance	Returns a list of the registered service instances on the service registry
createServiceInstance()	ServiceInstance newServiceInstance Service instance information being registered	anyURI	Creates a new individual service instance on the service registry. Return URI to the created service instance.

3.19 ServiceRegistrationROAInterface

Stereotype:
Type: Abstract Class
Extends: ROAServiceInterface
Package: fi.vtt.dsp.service.serviceregistry.roa

Main abstract class for the service registration ROA/REST interface

3.20 ServiceRegistrationResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Individual service registration item resource

Operations

Method	Params [type, name, notes]	Return values	Notes
getRegistration()		ServiceRegistryEntry	Returns a resource representation of the identified service registration
updateRegistration()	ServiceRegistryEntry updatedServiceRegistration Updated service registration	void	Updates the identified service's registrations
removeRegistration()		void	Removes the identified service registration from the service registry

3.21 ServiceRegistrationsResource

Stereotype: «Resource»
Type: Class
Extends: ItemCollectionResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Main collection resource for the service registrations

Operations

Method	Params [type, name, notes]	Return values	Notes
getListOfRegistrations()		ServiceRegistryEntry	Returns a list of the registered service registrations
createRegistration()	ServiceRegistryEntry registrationEntry Service registry entry being created	anyURI	Creates a new service registration to the service registry

3.22 TechnicalServiceDescriptionResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Resource for maintaining the technical service description in the identified service

Operations

Method	Params [type, name, notes]	Return values	Notes
getTechnicalServiceDescirption()		TechnicalService Description	Returns the technical service description for the identified service description
updateTechnicalServiceDescirption ()	TechnicalServiceDescriptio n updatedTechnicalDescripti on Updated technical service description information	void	Updates the technical service description in the identified service description
removeTechnicalServiceDescirptio n()		void	Removes the technical service description and corresponding service instance, if any, from the identified service description.

3.23 TechnicalServiceDescriptionsResource

Stereotype: «Resource»
Type: Class
Extends: ItemCollectionResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Main container resource for the technical service descriptions on the service description

Operations

Method	Params [type, name, notes]	Return values	Notes

getListOfTechnicalServiceDescriptions()		TechnicalServiceDescription	Returns a list of the technical service descriptions associated with the service description
createTechnicalServiceDescription()	TechnicalServiceDescription newTechnicalServiceDescription	anyURI	Creates a new technical service description for the service description. Returns URI to the created dependency.

3.24 UserFeedbackResource

Stereotype: «Resource»
Type: Class
Extends: ItemResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Individual resource for maintaining individual user feedback

Operations

Method	Params [type, name, notes]	Return values	Notes
readUserFeedback()		UserFeedback	Returns individual item representation of this resource
updateUserFeedback()	UserFeedback updatedUserFeedback Updated item presentation of the resource	anyURI	Updates this item resource with the provided content
removeUserFeedback()		void	Removes this item resource

3.25 UserFeedbacksResource

Stereotype: «Resource»
Type: Class
Extends: ItemCollectionResource
Package: fi.vtt.dsp.service.serviceregistry.roa

Main container resource for maintaining and managing user feedback in the given service description

Operations

Method	Params [type, name, notes]	Return values	Notes
createUserFeedback()	UserFeedback UserFeedback Representation of the resource to be added into the collection	anyURI	Creates and adds a new item into the collection. Returns URI to the created item resources.
readListOfUserFeedback()		UserFeedback	Reads and returns a list of the contained items

3.26 UserProfile

Stereotype: «Resource»
Type: Class
Extends:
Package: fi.vtt.dsp.service.serviceregistry.roa

Userprofile resource for registering, retrieving and removing user profiles utilized in the service registry.

Operations

Method	Params [type, name, notes]	Return values	Notes
DELETE()	string email	boolean	
GET()	string email	UserProfile	
CREATE()	UserProfile newProfile	boolean	
UPDATE()	UserProfile newUserProfile	boolean	

4 Appendix #2 Common service registry data

This appendix illustrates the data model of the service registry that is shared by the both styles of the service registry interfaces in the Figure 1. The Binding and ServiceAvailability classes are part of the common framework data package and defined elsewhere.

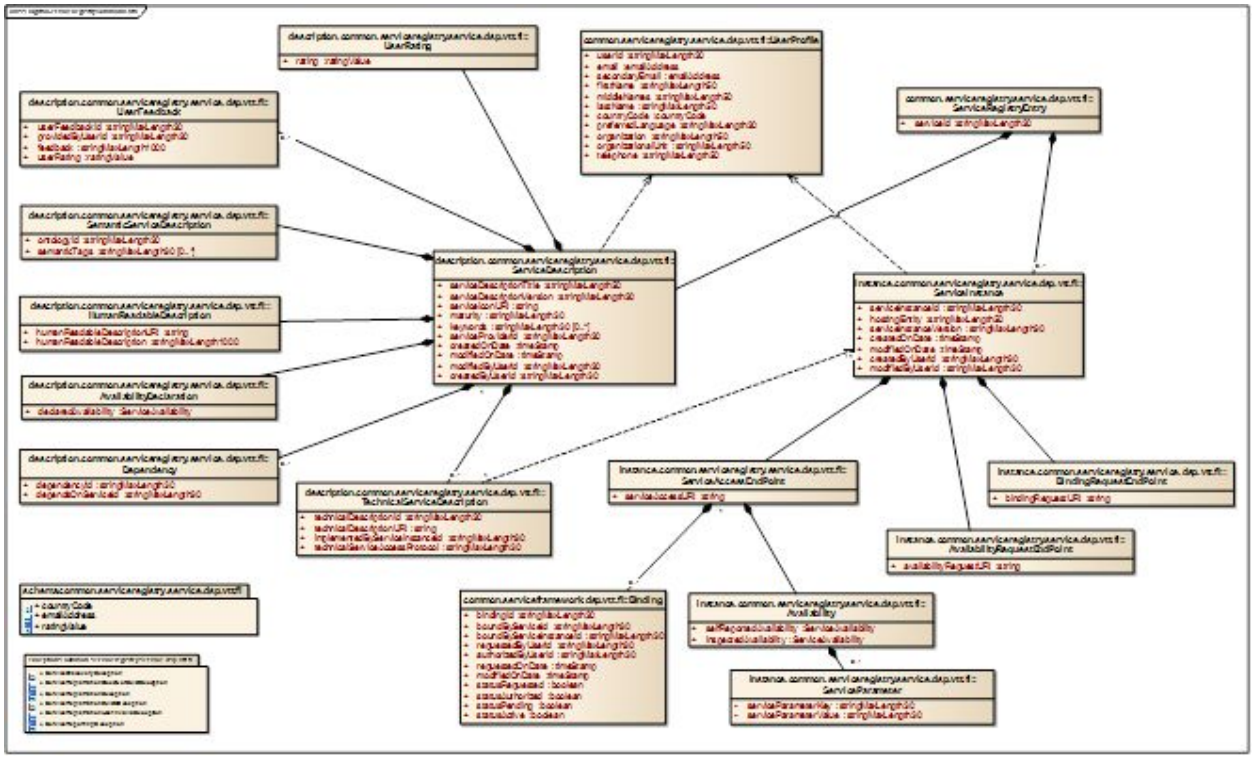

Figure 1. Service registry data model shared between the SOA and ROA style service interfaces

The following chapters give details of the data model components.

4.2 ServiceRegistryEntry

Stereotype:
Type: Class
Extends:
Package: common.serviceregistry.service.dsp.vtt.fi

Main entity of the registered service containing its description and a set of deployed service instances

Attributes

Attribute	Type	Constraints and tags	Notes

serviceId	stringMaxLength30	*Default:*	Service's unique identifier issued by the service registry

4.3 UserProfile

Stereotype:
Type: Class
Extends:
Package: common.serviceregistry.service.dsp.vtt.fi

UserProfile class describing user profiles used in the service registry.

Attributes

Attribute	Type	Constraints and tags	Notes
userId	stringMaxLength30	*Default:*	Unique user id.
email	emailAddress	*Default:*	User's email address.
secondaryEmail	emailAddress	*Default:*	User's secondary email address.
firstName	stringMaxLength50	*Default:*	User's first name.
middleNames	stringMaxLength50	*Default:*	User's middle names.
lastName	stringMaxLength50	*Default:*	User's last name.
countryCode	countryCode	*Default:*	User's 2 character country code.
preferredLanguage	stringMaxLength30	*Default:*	User's preferred language.
organization	stringMaxLength50	*Default:*	User's organization.
organizationalUnit	stringMaxLength50	*Default:*	User's organizational unit.
telephone	stringMaxLength50	*Default:*	User's telephone number.

4.4 AvailabilityDeclaration

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Main class for the declared availability by the developer for this service

Attributes

Attribute	Type	Constraints and tags	Notes
declaredAvailability	ServiceAvailability	*Default:*	Declared service availability

4.5 Dependency

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Dependencies that this service is depending on

Attributes

Attribute	Type	Constraints and tags	Notes
dependencyId	stringMaxLength30	*Default:*	Unique identifier of the depency
dependsOnServiceId	stringMaxLength30	*Default:*	Unique identifier of the service that is depended on

4.6 HumanReadableDescription

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Class for the human readable service description like web pages or other documentation

Attributes

Attribute	Type	Constraints and tags	Notes
humanReadableDescriptionURI	string	*Default:*	URI for the provided human readable description or documentation
humanReadableDescription	stringMaxLength1000	*Default:*	Provided human readable description or document content

4.7 SemanticServiceDescription

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Semantic description of the service. Reserved for future used.

Attributes

Attribute	Type	Constraints and tags	Notes
ontologyId	stringMaxLength30	*Default:*	
semanticTags [0..*]	stringMaxLength30	*Default:*	

4.8 ServiceDescription

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Main entity for the identified and registered service

Attributes

Attribute	Type	Constraints and tags	Notes
serviceDescriptionTitle	stringMaxLength50	*Default:*	Well descriptive title of the service description
serviceDescriptionVersion	stringMaxLength30	*Default:*	Version identifier of the service description following developer's own format
serviceIconURI	string	*Default:*	URI to the logo representing the service in the registry
maturity	stringMaxLength30	*Default:*	Maturity of the service description
keywords Collection [0..*]	stringMaxLength30	*Default:*	Selected set of keywords describing the service
serviceProviderId	stringMaxLength30	*Default:*	Identification of the service provider
createdOnDate	timeStamp	*Default:*	Date this service description was created
modifiedOnDate	timeStamp	*Default:*	Date this service description was modified last time
modifiedByUserId	stringMaxLength30	*Default:*	
createdByUserId	stringMaxLength30	*Default:*	

4.9 TechnicalServiceDescription

Stereotype:
Type: Class
Extends:

Package: description.common.serviceregistry.service.dsp.vtt.fi

Main class for the technical service description

Attributes

Attribute	Type	Constraints and tags	Notes
technicalDescriptionId	stringMaxLength30	*Default:*	Unique identifier of the technical service description
technicalDescriptionURI	string	*Default:*	URI to the service description
implementedByServiceInstanceId	stringMaxLength30	*Default:*	Service instance identifier of the service that implements this technical description
technicalServiceAccessProtocol	stringMaxLength30	*Default:*	Service access protocol identifier intended to be used with the service technically described

4.10 UserFeedback

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Main class for user provided feedback for the service

Attributes

Attribute	Type	Constraints and tags	Notes

userFeedbackId	stringMaxLength30	*Default:*	Unique identifier of the user feed back
providedByUserId	stringMaxLength30	*Default:*	User id of the feedback provider
feedback	stringMaxLength1000	*Default:*	Free text formed feedback by the user
userRating	ratingValue	*Default:*	Rating value provided by the user for the service

4.11 UserRating

Stereotype:
Type: Class
Extends:
Package: description.common.serviceregistry.service.dsp.vtt.fi

Class for the service user based ratings

Attributes

Attribute	Type	Constraints and tags	Notes
rating	ratingValue	*Default:*	User provided average rating value

4.12 ServiceDiscoveryException

Stereotype:
Type: Class
Extends: ServiceException
Package: exceptions.common.serviceregistry.service.dsp.vtt.fi

Exception thrown when service discovery operation fails. Cause object may provide detailed information about the nature of the exception.

4.13 ServiceRegistrationDoesNotExistException

Stereotype:

Type: Class
Extends: ServiceException
Package: exceptions.common.serviceregistry.service.dsp.vtt.fi

Exception thrown if the targeted service description or instance registration does not exist. Cause object may provide detailed information about the nature of the exception.

4.14 ServiceRegistrationException

Stereotype:
Type: Class
Extends: ServiceException
Package: exceptions.common.serviceregistry.service.dsp.vtt.fi

Exception thrown if service description or instance registration operation fails. Cause object may provide detailed information about the nature of the exception.

4.15 ServiceRegistrationExistsException

Stereotype:
Type: Class
Extends: ServiceException
Package: exceptions.common.serviceregistry.service.dsp.vtt.fi

Exception thrown if the provided service instance or description is already registered in the system. Cause object may provide detailed information about the nature of the exception.

4.16 ServiceRegistrationUserInvalidException

Stereotype:
Type: Class
Extends: ServiceException
Package: exceptions.common.serviceregistry.service.dsp.vtt.fi

Exception thrown if the provided user information is invalid or does not exist in the user profiles. Cause object may provide detailed information about the nature of the exception.

4.17 ServiceReportingException

Stereotype:
Type: Class

Extends: ServiceException
Package: exceptions.common.serviceregistry.service.dsp.vtt.fi

Exception thrown if availability or binding reporting operation fails. Cause object may provide detailed information about the nature of the exception.

4.18 Availability

Stereotype:
Type: Class
Extends:
Package: instance.common.serviceregistry.service.dsp.vtt.fi

Main class for holding service instance's self reported or inspected availability information and monitored service parameters

Attributes

Attribute	Type	Constraints and tags	Notes
selfReportedAvailability	ServiceAvailability	*Default:*	Availability information reported by the service instance itself
inspectedAvailability	ServiceAvailability	*Default:*	Service information retrieved and inspected from the service by service registry's availability monitoring component

4.19 AvailabilityRequestEndPoint

Stereotype:
Type: Class
Extends:
Package: instance.common.serviceregistry.service.dsp.vtt.fi

Main class for holding service instance's availability request endpoint information

Attributes

Attribute	Type	Constraints and tags	Notes

availabilityRequestURI	string	*Default:*	Availability request URI of the service instance for requesting availability information. Service instance can implement availability information request endpoint or report the information by itself.

4.20 BindingRequestEndPoint

Stereotype:
Type: Class
Extends:
Package: instance.common.serviceregistry.service.dsp.vtt.fi

Main class for holding service instance's binding request endpoint information

Attributes

Attribute	Type	Constraints and tags	Notes
bindingRequestURI	string	*Default:*	Binding URI of the service instance for requesting binding information. Service instance can implement binding information request endpoint or report the information by itself

4.21 ServiceAccessEndPoint

Stereotype:
Type: Class
Extends:
Package: instance.common.serviceregistry.service.dsp.vtt.fi

Main class for holding service instance's service access endpoint information

Attributes

Attribute	Type	Constraints and tags	Notes
serviceAccessURI	string	*Default:*	URI for the service instance for binding and service access

4.22 ServiceInstance

Stereotype:
Type: Class
Extends:
Package: instance.common.serviceregistry.service.dsp.vtt.fi

Main class holding information for the service instance in the service registry

Attributes

Attribute	Type	Constraints and tags	Notes
serviceInstanceId	stringMaxLength30	*Default:*	Unique identifier for the service instance that implements the service identified by service id.
hostingEntity	stringMaxLength50	*Default:*	Organizational name of the entity hosting the service instance
serviceInstanceVersion	stringMaxLength30	*Default:*	Version of the provided service instance
createdOnDate	timeStamp	*Default:*	Date this service instance was registered
modifiedOnDate	timeStamp	*Default:*	Date this service instance was modified last time
createdByUserId	stringMaxLength30	*Default:*	

modifiedByUserId	stringMaxLength30	*Default:*	

4.23 ServiceParameter

Stereotype:
Type: Class
Extends:
Package: instance.common.serviceregistry.service.dsp.vtt.fi

Remotely monitored service parameters as key-value pairs

Attributes

Attribute	Type	Constraints and tags	Notes
serviceParameterKey	stringMaxLength30	*Default:*	
serviceParameterValue	stringMaxLength30	*Default:*	

4.24 countryCode

Stereotype: «string»
Type: Class
Extends: string
Package: schema.common.serviceregistry.service.dsp.vtt.fi

Service registry schema type for country codes. Maximum length is 2 characters.

4.25 emailAddress

Stereotype: «string»
Type: Class
Extends: string
Package: schema.common.serviceregistry.service.dsp.vtt.fi

Service registry common schema type for email addresses

4.26 ratingValue

Stereotype:	«int»
Type:	Class
Extends:	string
Package:	schema.common.serviceregistry.service.dsp.vtt.fi

Service registry schema type for rating values ranging from 0 to 100.

5 Appendix #3 ServiceData ontology

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
	<!ENTITY dct "http://purl.org/dc/terms/" >
	<!ENTITY dcat "http://www.w3.org/ns/dcat#" >
	<!ENTITY foaf "http://xmlns.com/foaf/0.1/" >
	<!ENTITY owl "http://www.w3.org/2002/07/owl#" >
	<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
	<!ENTITY qb "http://purl.org/linked-data/cube#" >
	<!ENTITY ssn "http://purl.oclc.org/NET/ssnx/ssn#" >
	<!ENTITY skos "http://www.w3.org/2004/02/skos/core#" >
	<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
	<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://profile.vtt.fi/ontologies/dhr/ServiceData#"
	xml:base="http://profile.vtt.fi/ontologies/dhr/ServiceData"
	xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
	xmlns:dcat="http://www.w3.org/ns/dcat#"
	xmlns:foaf="http://xmlns.com/foaf/0.1/"
	xmlns:dct="http://purl.org/dc/terms/"
	xmlns:owl="http://www.w3.org/2002/07/owl#"
	xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
	xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#"
	xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
	xmlns:qb="http://purl.org/linked-data/cube#"
	xmlns:skos="http://www.w3.org/2004/02/skos/core#">
	<owl:Ontology rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData">
	<owl:imports rdf:resource="http://purl.oclc.org/NET/ssnx/ssn"/>
	<owl:imports rdf:resource="http://purl.org/dc/terms/"/>
	<owl:imports rdf:resource="http://purl.org/linked-data/cube"/>
	<owl:imports rdf:resource="http://www.w3.org/ns/dcat"/>
	</owl:Ontology>

	<!--
	///////////////////////////////////////////////////////////////////////////////////
	//
	// Object Properties
	//
	///////////////////////////////////////////////////////////////////////////////////
	-->
```

```
        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#schemaID -->

        <owl:ObjectProperty rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#schemaID">
        <skos:definition>Link to a schema that is used when creating the service data description
</skos:definition>
        <rdfs:domain rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#ServiceDataDescription"/>
        </owl:ObjectProperty>


        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#serviceDataDescriptionURI -->

        <owl:ObjectProperty
rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#serviceDataDescriptionURI">
        <skos:definition>Link to a document that describes the service data description based on the defined
schema </skos:definition>
        <rdfs:domain rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#ServiceDataDescription"/>
        </owl:ObjectProperty>


        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#serviceSchema -->

        <owl:ObjectProperty rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#serviceSchema">
        <skos:definition>Link to the original schema of the service </skos:definition>
        <rdfs:domain rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#ServiceDataDescription"/>
        </owl:ObjectProperty>


        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#taxonomy -->

        <owl:ObjectProperty rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#taxonomy">
        <skos:definition>Link to taxonomy that is used to describe data of a service </skos:definition>
        <rdfs:domain rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#ServiceDataDescription"/>
        </owl:ObjectProperty>


        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#technicalDescriptionURI -->

        <owl:ObjectProperty
rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#technicalDescriptionURI">
        <skos:definition>Link (URI) to the technical description document </skos:definition>
        <rdfs:domain rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#Distribution"/>
        </owl:ObjectProperty>

        <!--
        ///////////////////////////////////////////////////////////////////////////////////
        //
        // Data properties
```

```
        //
        ///////////////////////////////////////////////////////////////////////////////
        -->


        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#serviceDataType -->

        <owl:DatatypeProperty rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#serviceDataType">
        <skos:definition>input or output data (sink or source)
</skos:definition>
        <rdfs:domain rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#Dataset"/>
        <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>

        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#purpose -->

 <owl:DatatypeProperty rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#purpose">
        <skos:definition>Defines the purpose for which a sink uses the data. This is defined only for a sink.
</skos:definition>
        <rdfs:domain rdf:resource="http://profile.vtt.fi/ontologies/dhr/ServiceData#Dataset"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:DatatypeProperty>


        <!--
        ///////////////////////////////////////////////////////////////////////////////
        //
        // Classes
        //
        ///////////////////////////////////////////////////////////////////////////////
        -->


        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#Dataset -->

        <owl:Class rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#Dataset">
        <rdfs:subClassOf rdf:resource="&dcat;Dataset"/>
        </owl:Class>


        <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#ServiceDataDescription -->

        <owl:Class rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#ServiceDataDescription">
        <rdfs:subClassOf rdf:resource="&dcat;Catalog"/>
        </owl:Class>


 <!-- http://profile.vtt.fi/ontologies/dhr/ServiceData#Distribution -->
```

```
        <owl:Class rdf:about="http://profile.vtt.fi/ontologies/dhr/ServiceData#Distribution">
        <rdfs:subClassOf rdf:resource="&dcat;Distribution"/>
        </owl:Class>


        <!-- http://purl.oclc.org/NET/ssnx/ssn#Observation -->

<owl:Class rdf:about="&ssn;Observation">
        <rdfs:subClassOf rdf:resource="&qb;Observation"/>
        </owl:Class>


        <!-- http://purl.org/linked-data/cube#ComponentProperty -->

        <owl:Class rdf:about="&qb;ComponentProperty">
        <rdfs:subClassOf rdf:resource="&ssn;Property"/>
        </owl:Class>


        <!-- http://purl.org/linked-data/cube#DataSet -->

        <owl:Class rdf:about="&qb;DataSet">
        <rdfs:subClassOf rdf:resource="&foaf;Document"/>
        </owl:Class>


        <!-- http://purl.org/linked-data/cube#Observation -->

        <owl:Class rdf:about="&qb;Observation"/>


        <!-- http://www.w3.org/ns/dcat#Dataset -->

 <owl:Class rdf:about="&dcat;Dataset">
        <rdfs:subClassOf rdf:resource="&qb;DataSet"/>
        </owl:Class>

</rdf:RDF>
```