Title:
A5.3-D3 HUMBOLDT Commons Specification / Framework Common Data Model V3
Author(s)/Organisation(s):
Ana Belén Antón / ETRA, Daniel Fitzner / FHG
Working Group:
Architecture Team/WP05
References:
A5.2-D3 [3.0] A Lightweight Introduction to the HUMBOLDT Framework V3
A5.2-D3 [3.1] Specification Introduction and Overview V3
A5.2-D3 [3.2] Mediator Service Component Specification
A5.2-D3 [3.3] Conceptual Schema Specification and Mapping
A5.2-D3 [3.3.1] The HUMBOLDT Alignment Editor
A5.2-D3 [3.4] Context Service Specification
A5.2-D3 [3.5] Workflow Design and Construction Service Specification
A5.2-D3 [3.6] Processing Components General Model and Implementations
A5.2-D3 [3.7] Information Grounding Service Component Specification
Quality Assurance:
☑ Review WP Leader: Thorsten Reitz (FhG)
☐ Review dependent WP leaders:
☐ Review Executive Board:
☑ Review others: Moses Gone (FhG)
Delivery Date: **30.11.2009**

Short Description:
This document describes the specification of the HUMBOLDT Commons as part of the HUMBOLDT software framework. For an overview of the entire framework v3.0 description, please refer to the main specification documents A5.2-D3 [3.0] and [3.1]. This component is not an actual service component like the other HUMBOLDT components which satisfy functional requirements, but represents the common data model. Therefore HUMBOLDT Commons integrates all core data structures used in more than one HUMBOLDT component.

Keywords:

Framework specification, logical architecture, physical architecture, common data structures.

History:

Version	Author(s)	Status	Comment
001	Ana Belén Antón	New	Initial specification of framework V2.0, template
002	Ana Belén Antón	rfc	New Commons after some requests
003	Ana Belén Antón	Rfc	Changes according to the ContextService and DQME specifications and requests. Diagrams updated.
004	Ana Belén Antón	Rfc	Updating the Transformer and Workflow description. Minor editing changes.
005	Daniel Fitzner	rfc	Updating the Context, MCR and Workflow models
006	Daniel Fitzner	rfc	Added sections on constraint handling

Table of contents

Figures

Tables

1 Introduction

1.1 Purpose of this document

This component is not an actual service component like the other HUMBOLDT services which satisfy functional requirements, but represents their common Data Model. The HUMBOLDT Commons integrates all core data structures used in more than one HUMBOLDT component for information exchange. Therefore only the *Information Viewpoint* joint to the *Introduction* will be available in this specification document.

1.2 Abbreviations and Definitions used in this document

This section summarizes all abbreviations which are used specifically for this document. It collects several kinds of abbreviations to provide a single point of reference, including names of modules, protocols, services, standards and tools. If any general abbreviation is not found here, please see Deliverable A5.2-D3.

Abbrev.	***Name***	***Definition***
CS	Context Service	Service component of the HUMBOLDT Framework in charge of the management and provision of the specific contexts. This component consists of two modules: the Context Management Service and the User Management Service. See the document A5.2-D3 [3.4]
DQME	Data Quality Measurement and Evaluation	Service component of the HUMBOLDT Framework in charge of the measurement and evaluation of the data quality. See the document A5.2-D2 [2.7]
MCR	Mediator Complex Request	The MediatorComplexRequest is a data structure used internally by the Mediator Service. It is basically a container for the Constraints and Parameters collected from a User's request by the Context Service.
MS	Mediator Service	Service component of the HUMBOLDT Framework in charge of the management of the actual execution of a transformation and to provide the user with some support in specifying his product definition. See the document A5.2-D3 [3.2]
WCS	Web Coverage Service	International standard defined by the OGC for providing interoperable access to geospatial coverages, such as satellite images, digital aerial photos, digital elevation data, and other phenomena represented by values at each measurement point.
WDCS	Workflow Design and Construction Service	Service component of the HUMBOLDT Framework in charge of the creation of geospatial workflows. See the document A5.2-D3 [3.5]
WFS	Web Feature	International standard for web services offering geospatial

Abbrev.	*Name*	*Definition*
	Service	feature data.
WMS	Web Map Service	International standard for web services offering georeferenced maps.
WPS	Web Processing Service	International standard for web services offering geoprocessing functionality.

Table 1: Abbreviations used in the description of the component

This section gives a few definitions valid for the context of this document. As in the section before, if any general definition is not found here, please see Deliverable A5.2-D3.

Context	A Context is defined as the specific user requirements related to the environment in which his/her business operates when they are retrieving geo-information.
Grounding	Groundings represent the geospatial information resources distributed on the Internet, which follow the OGC standards such as WFS, WMS.
Transformer	A Transformer is the HUMBOLDT term for geoprocessing functionality / components.
Workflow	A composition of transformers and groundings whose execution can answer requests for geospatial data that can not be answered by any of the known data sources.

Table 2: Definitions used in the description of the component

2 Information Viewpoint

This chapter gives the information viewpoint of the HUMBOLDT Framework v3.0. This covers various data structures that are being used for storage and exchange between some of the HUMBOLDT components defined in that version. In the Annex, the different exchange models can be found as XML-schema.

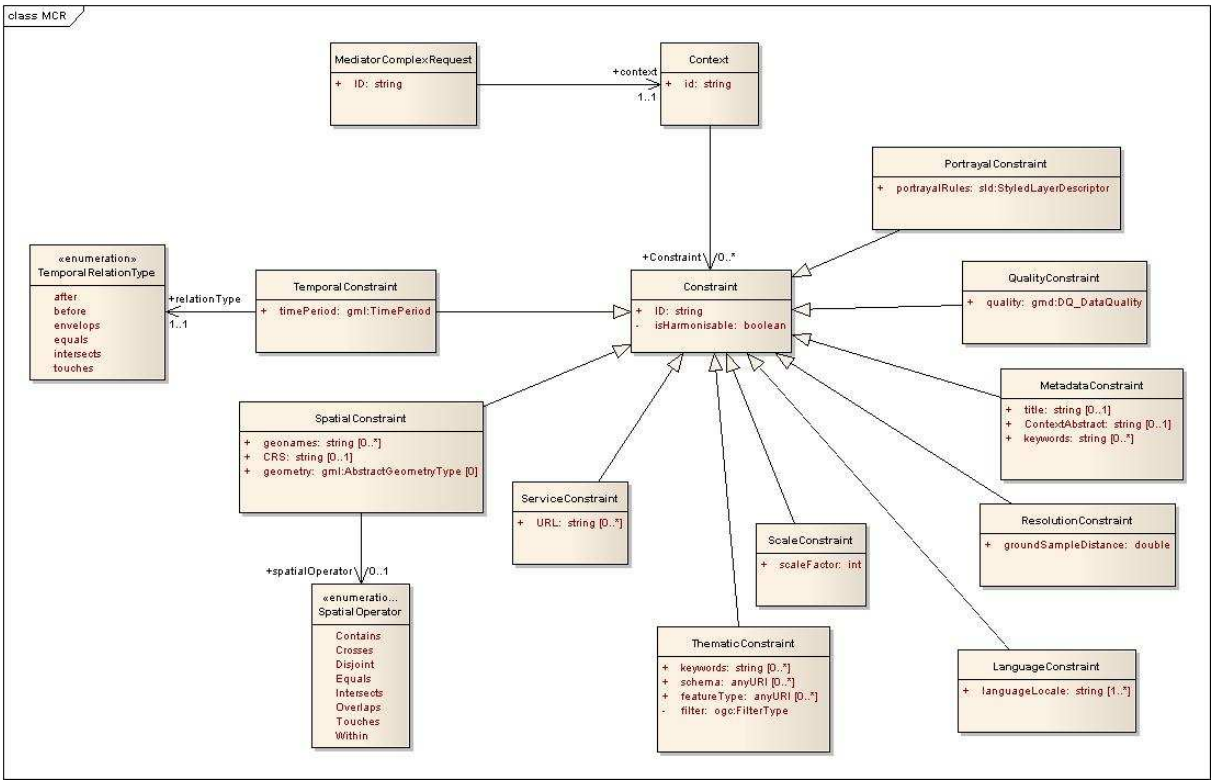


Figure 1: The HUMBOLDT Constraint Model

2.1 Mediator Complex Request

The Mediator Complex Request (MCR) represents a combination of a user's query, as issued to the HUMBOLDT Mediator Service in the form of a standard OGC request, and his context as specified in the HUMBOLDT Context Service. Thus, the MCR is the main message structure used in the HUMBOLDT framework when it comes to expressing requests. It is synthesized from the different logical and physical schemas used to express requests in the most common interface specification in the geospatial domain, such as WFS, WMS, WPS and WCS, and is enriched by information from a Context document.

The MCR has the following key elements that define it:

- *A unique identifier:* This identification is used throughout the HUMBOLDT system to always be able to track back which request led to the creation of a sub-request or response. It can be used for authentication and communication aspects, such as relaying back the answer to the original requester.
- *A Context:* A context consists of a set of constraints. Each Constraint within a context can be understood as a rule that the result to be returned to the requesting client has to fulfill. There are very different types of Constraints as described in section 2.3.

The constraints as derived from the users query directly (e.g. a WFS / WMS query) are used to enrich the context. However, the original user context as hold and maintained within the context service is left unchanged and can always be derived in its original form.

2.2 Context

A *Context* in HUMBOLDT is used to express constraints on geospatial data sets. Contexts are maintained and administrated in the *HUMBOLDT Context Service*. Further, contexts are exchanged between the *HUMBOLDT Context Service* and the *HUMBOLDT Mediator Service*.

Contexts consist of a set of constraints of different types (for a detailed description, see next section), e.g. spatial constraints, portrayal constraints etc. Constraints of different types are handled within the framework as being connected by **conjunction** (i.e. logical AND). Further, users are allowed to specify multiple constraints of the same type, such as multiple temporal constraints. These are handled by the framework as being connected by **disjunction** (logical OR). Hence, we do not allow complex logical expressions when building contexts since this is not required within HUMBOLDT.

The constraint model as specified in section 2.3 is derived from HUMBOLDT scenario requirements as given in the document *A5.2-D3 [3.4] HUMBOLDT Context Service Specification*.

2.3 The HUMBOLDT Constraint Model

A **constraint** expresses restrictions on the data to be returned, expressed by users. Constraints are persisted as part of contexts in the Context Service (CS), and are used in the Mediator Service (MS), the Workflow Design and Construction Service (WDCS) and the Information Grounding Service (IGC). What follows is a short introduction to the HUMBOLDT constraint model.

2.3.1 Temporal Constraint

Using database terminology, this constraint refers to the ***valid time*** of the data instead of the time of creation of the data set (i.e. the transaction time). This means, it refers to the time period the real world objects represented by the data have been valid / existing in the real world. For example, in case a user requests data on rivers in europe at the time of the middle ages, he would specify the time period he is interested in (i.e. the middle ages) within this constraint and not the time, the data set was created. Sometimes, the valid time and the transaction time can be equal or close to each other, e.g. for satellite images. As a reference system, the Gregorian calendar with UTC shall be used.

Temporal Constraint Handling

Those data source will be included that satisfy the temporal requirements as imposed by this constraint.

2.3.2 Spatial Constraint

A Spatial Constraint can be used to express a region of interest or a spatial reference system the data must be specified in. The region of interest can either be expressed via *toponyms* or *placenames* or via a rectangular bounding box (or both). The bounding box shall be specified using the WGS84 reference system. The constraint on the spatial reference system shall be specified in the form: “EPSG:<*epsg-code*>. The spatial operators are only used, in case an envelope is specified.

Spatial Constraint Handling

The spatial constraint is subject to automated harmonisation within the framework. In case, the constraint specifies a spatial reference system those data sources are included in the result that directly serve the data specified in the requested spatial reference system or that can be transformed such that the resulting data set satisfies the constraint.

In case, a *geometry* and a *spatial operator* is specified, those data sets are included whose bounding boxes satisfy the spatial operator / geometry. Not only single data sets are taken into consideration here but also multiple. For example, in case the geometry covers an area that is not served by a single data source but by multiple, the combination (spatial merge / union) of these multiple data sets is taken into consideration in the result.

2.3.3 Service Constraint

The service constraint allows users to specify the URL of a specific service, e.g. a specific WFS they want to use.

Service Constraint Handling

In case, a service URL is specified, only those services that are accessible via that URL are included in the result set.

2.3.4 Language Constraint

The Language Constraint allows the user to express his preferences for the natural language that the data sets should be presented in. Language constraints refer to the data schema (e.g. application schema) and therefore Feature Type names and attributes as well as to values of attributes. For this purpose, ISO 639-2 and ISO 639-3 language codes shall be used.

Language Constraint Handling

The language constraint is closely related to the thematic constraint, since application schemas are always specified in a certain language. However, the language constraint in HUMBOLDT mainly refers to attribute values. Hence, all data sources are included in the result that conform to the language specified in the language constraint directly or that can be transformed (i.e. the attribute values) such that they satisfy the request.

2.3.5 Thematic Constraint

The thematic constraint allows users to specify keywords (e.g. *water*, *soil*) related to data they are interested in. Further, in enables users to directly specify a schema (or a set of schemas) and feature types. Additionally, the thematic constraint holds a standard *OGC Filter Encoding* filter, which can be used to filter based on attribute values.

Thematic Constraint Handling

In case, only a set of keywords is set within a thematic constraint, information on all data sets that contain such keywords in their metadata descriptions are returned. In case, schemas are specified, only information on those data sets that conform to that schema directly or that can be transformed via a schema transformation is returned. The same applies to Feature Types.

2.3.6 Scale Constraint

Digital data usually does not have an inherent scale since scale is always related to display. However, the level of detail e.g. of vector data might have a maximum scale to which it is suitable to display. For example, in case a vector data set is digitized from a 1:50 000 map e.g. of highways, its level of details or accuracy is not suitable for displaying / printing it as a 1 : 20 000 map.

Scale Constraint Handling

The scale constraint allows users to specify the minimum scale they are interested in. For example, if they specify a scale of *scaleFactor = 50 000*, i.e. 1 : 50 000, those data sets match the scale whose scale is larger. For example, a data set with a scale 1 : 25 000 matches while a data set with scale 1 : 200 000 does not.

2.3.7 Resolution Constraint

The resolution constraint refers to the *Ground Sample Distance* of a raster / coverage data set, i.e. the distance on the earth surface covered by the height / width of a pixel. For example, in a satellite image with a ground sample distance of 10 meters, one single pixel covers the area of 10 m². The unit for the Ground Sample Distance shall be meters.

Resolution Constraint Handling

Similar to the scale, those data sets are included in the answer set to the request that are more detailed, i.e. where the *Ground Sample Distance* is smaller than the one specified in the constraint. The *Ground Sample Distance* has no relevance for vector data, e.g. data delivered by WFS. However, in case the user requests vector data e.g. from a WFS to be portrayed within the framework, the resolution constraint might be used as a parameter in the rasterization process performed within the HUMBOLDT Mediator Service.

2.3.8 Quality Constraint

This constraint allows the definition of quality parameters that the data in question will need to fulfill. For this, ISO 19113 quality definitions have been used.

Quality Constraint Handling

Those data source will be included that satisfy the quality requirements as imposed by this constraint.

2.3.9 Metadata Constraint

This type of Constraint allows testing for conformance to other metadata elements than those which have explicit constraint types, such as responsible party or certain identification such as keywords.

The current scope of this specification covers the following metadata types to be filtered:

- Title (ISO19139: title)
- ContextAbstract (ISO19139: abstract)
- Keywords (ISO19139: keywords)

Metadata Constraint Handling

Those data source will be included that satisfy the metadata requirements as imposed by this constraint.

2.3.10 Portrayal Constraint

This constraint holds an OGC Styled Layer Descriptor Document that specifies the requested styling. Usually, such styling is tailored to specific *FeatureTypes* of specific schemas. Hence, the feature types appearing within the portrayal constraint must also appear in the list of requested feature types in the thematic constraint.

Portrayal Constraint Handling

The portrayal constraint is handled differently in the framework, depending on the data source that serves the requested *Feature Types*. For WFS, the SLD is used for portraying the data within the framework (precisely: within the HUMBOLDT Mediator Service). In case of WMS, the SLD is directly applied on the individual service, in case it supports user defined styling.

2.4 Workflow Exchange Model

Figure 4 shows the model for workflow exchange between WDCS and Mediator Service. Transformers are either *WPSTransformer*, in case the transformation is encapsulated within an external WPS. In this case, the Workflow Service sends the *WPS URL* and the *ProcessIdentifier* as it appears in the *GetCapabilities* response of that WPS. Or, Transformers are implemented in java. In this case, the WDCS just sends the name of the java class that holds the transformation algorithm. Additionally, for each transformer, a mapping of the names of the inputs is passed. This mapping maps a parameter name either to:

- a value, in case of literal inputs
- a pointer to a data source, e.g. a WFS request
- a pointer to another process in the set of processes passed from the WDCS to the MS

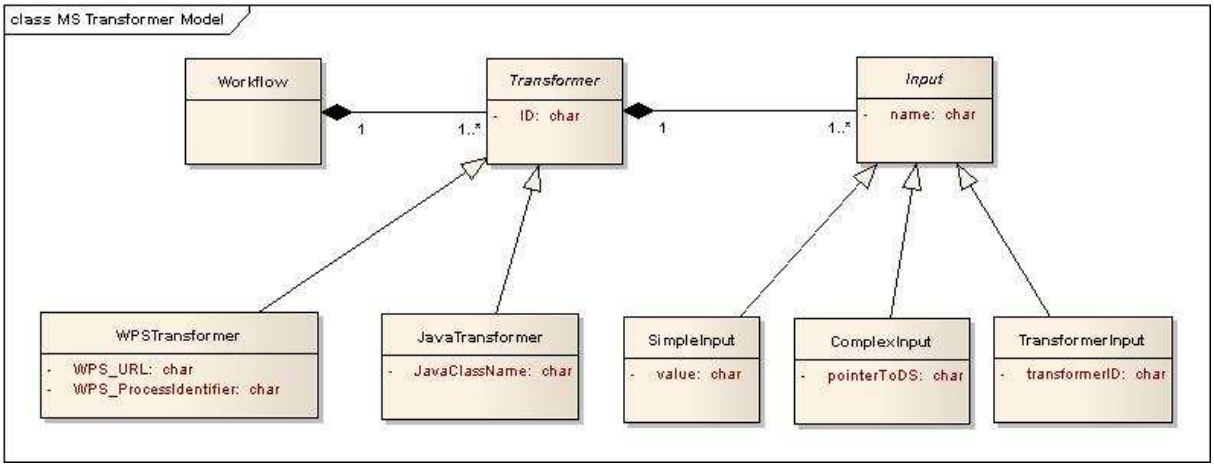
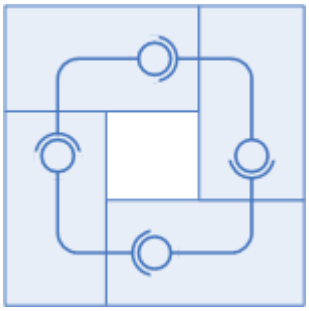


Figure 2: Workflow Exchange Model

3 Summary & Outlook



This specification provided an overview of the Information Viewpoint of the HUMBOLDT Framework. It is the data structures shared by different services defined in HUMBOLDT.

In previous versions of the specification, this “component” was not defined, but the lessons learned in the version 1.0 of the specification have demonstrated that common structures must be organized in order to share information in a common manner within the HUMBOLDT Framework.

Annex

Annex A – The Workflow Exchange Model (XML-schema)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://esdi-humboldt.eu/schemas/workflowexchange">
      <xs:element name="Workflow" type="WorkflowType"/>
      <xs:complexType name="WorkflowType">
            <xs:sequence>
                  <xs:element name="Transformer" type="TransformerType"
                  maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="Transformer" type="TransformerType"/>
      <xs:complexType name="Transformer" abstract="true">
            <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="Input" type="InputType"
                  maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="WPSTransformer" type="WPSTransformer"/>
      <xs:complexType name="WPSTransformerType">
            <xs:complexContent>
                  <xs:extension base="TransformerType">
                        <xs:sequence>
                              <xs:element name="WPS_URL" type="xs:string"/>
                              <xs:element name="WPS_ProcessIdentifier"
                              type="xs:string"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="JavaTransformer" type="JavaTransformerType"/>
      <xs:complexType name="JavaTransformerType">
            <xs:complexContent>
                  <xs:extension base="TransformerType">
                        <xs:sequence>
                              <xs:element name="JavaClassName"
                              type="xs:string"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="Input" type="InputType"/>
      <xs:complexType name="InputType" abstract="true">
            <xs:sequence>
                  <xs:element name="name" type="xs:string"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="SimpleInput" type="SimpleInputType"/>
      <xs:complexType name="SimpleInputType">
            <xs:complexContent>
                  <xs:extension base="InputType">
                        <xs:sequence>
                              <xs:element name="value" type="xs:string"/>
                        </xs:sequence>
```

```
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="ComplexInput" type="ComplexInputType"/>
        <xs:complexType name="ComplexInputType">
            <xs:complexContent>
                <xs:extension base="InputType">
                    <xs:sequence>
                        <xs:element name="pointerToDS"
                        type="xs:string"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="TransformerInput" type="TransformerInputType"/>
        <xs:complexType name="TransformerInputType">
            <xs:complexContent>
                <xs:extension base="InputType">
                    <xs:sequence>
                        <xs:element name="transformerID"
                        type="xs:string"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
</xs:schema>
```

Annex B – The Context Exchange Model (XML-Schema)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:sld="http://www.opengis.net/sld"
xmlns:gml="http://www.opengis.net/gml/3.2" targetNamespace="http://esdi-
humboldt.eu/schemas/context">
        <xs:import namespace="http://www.opengis.net/sld"/>
        <xs:import namespace="http://www.opengis.net/gml"/>
        <xs:import namespace="http://www.isotc211.org/2005/gmd"/>
        <xs:import namespace="http://www.opengis.net/ogc"/>
        <xs:element name="TemporalConstraint" type="TemporalConstraintType"/>
        <xs:complexType name="TemporalConstraintType">
            <xs:complexContent>
                <xs:extension base="ConstraintType">
                    <xs:sequence minOccurs="0" maxOccurs="*">
                        <xs:element name="timePeriod"
                        type="gml:TimePeriod" maxOccurs="1"/>
                        <xs:element name="relationType"
                        type="TemporalRelationType"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="SpatialConstraint" type="SpatialConstraintType"/>
        <xs:complexType name="SpatialConstraintType">
            <xs:complexContent>
                <xs:extension base="ConstraintType">
```

```
                        <xs:sequence minOccurs="0" maxOccurs="*">
                              <xs:element name="geonames" type="xs:string"
                              minOccurs="0" maxOccurs="unbounded"/>
                              <xs:element name="CRS" type="xs:string"
                              minOccurs="0" maxOccurs="1"/>
                              <xs:element name="geometry"
                              type="gml:AbstractGeometryType" minOccurs="0"
                              maxOccurs="unbounded"/>
                              <xs:element name="spatialOperator"
                              type="SpatialOperator" minOccurs="0"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="ServiceConstraint" type="ServiceConstraintType"/>
      <xs:complexType name="ServiceConstraintType">
            <xs:complexContent>
                  <xs:extension base="Constraint">
                        <xs:sequence minOccurs="0" maxOccurs="*">
                              <xs:element name="URL" type="xs:string"
                              minOccurs="0" maxOccurs="unbounded"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:simpleType name="TemporalRelationType">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="after"/>
                  <xs:enumeration value="before"/>
                  <xs:enumeration value="envelops"/>
                  <xs:enumeration value="equals"/>
                  <xs:enumeration value="intersects"/>
                  <xs:enumeration value="touches"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:element name="ThematicConstraint" type="ThematicConstraintType"/>
      <xs:complexType name="ThematicConstraint">
            <xs:complexContent>
                  <xs:extension base="ConstraintType">
                        <xs:sequence minOccurs="0" maxOccurs="1">
                              <xs:element name="keywords" type="xs:string"
                              minOccurs="0" maxOccurs="unbounded"/>
                              <xs:element name="schema" type="xs:anyURI"
                              minOccurs="0" maxOccurs="unbounded"/>
                              <xs:element name="featureType"
                              type="xs:anyURI" minOccurs="0"
                              maxOccurs="unbounded"/>
                              <xs:element name="filter"
                              type="ogc:FilterType"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="LanguageConstraint" type="LanguageConstraintType"/>
      <xs:complexType name="LanguageConstraintType">
            <xs:complexContent>
                  <xs:extension base="Constraint">
                        <xs:sequence minOccurs="0" maxOccurs="1">
                              <xs:element name="languageLocale"
                              type="xs:string" maxOccurs="unbounded"/>
```

```
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
            <xs:element name="ResolutionConstraint"
            type="ResolutionConstraintType"/>
        <xs:complexType name="ResolutionConstraint">
            <xs:complexContent>
                <xs:extension base="ConstraintType">
                    <xs:sequence minOccurs="0" maxOccurs="1">
                        <xs:element name="groundSampleDistance"
                        type="xs:double"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="ScaleConstraint" type="ScaleConstraintType"/>
        <xs:complexType name="ScaleConstraint">
            <xs:complexContent>
                <xs:extension base="ConstraintType">
                    <xs:sequence minOccurs="0" maxOccurs="1">
                        <xs:element name="scaleFactor" type="xs:int"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
            <xs:element name="PortrayalConstraint"
            type="PortrayalConstraintType"/>
        <xs:complexType name="PortrayalConstraint">
            <xs:complexContent>
                <xs:extension base="ConstraintType">
                    <xs:sequence minOccurs="0" maxOccurs="1">
                        <xs:element name="portrayalRules"
                        type="sld:StyledLayerDescriptor"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:element name="MetadataConstraint" type="MetadataConstraintType"/>
        <xs:complexType name="MetadataConstraint">
            <xs:complexContent>
                <xs:extension base="ConstraintType">
                    <xs:sequence minOccurs="0" maxOccurs="1">
                        <xs:element name="title" type="xs:string"
                        minOccurs="0"/>
                        <xs:element name="ContextAbstract"
                        type="xs:string" minOccurs="0"/>
                        <xs:element name="keywords" type="xs:string"
                        minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:simpleType name="SpatialOperator">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Contains"/>
                <xs:enumeration value="Crosses"/>
                <xs:enumeration value="Disjoint"/>
                <xs:enumeration value="Equals"/>
                <xs:enumeration value="Intersects"/>
```

```
                  <xs:enumeration value="Overlaps"/>
                  <xs:enumeration value="Touches"/>
                  <xs:enumeration value="Within"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:element name="Constraint" type="ConstraintType"/>
      <xs:complexType name="Constraint">
            <xs:sequence>
                  <xs:element name="ID" type="xs:string"/>
                  <xs:element name="isHarmonisable" type="xs:boolean"/>
            </xs:sequence>
      </xs:complexType>
      <xs:element name="QualityConstraint" type="QualityConstraintType"/>
      <xs:complexType name="QualityConstraint">
            <xs:complexContent>
                  <xs:extension base="ConstraintType">
                        <xs:sequence>
                              <xs:element name="quality"
                              type="gmd:DQ_DataQuality"/>
                        </xs:sequence>
                  </xs:extension>
            </xs:complexContent>
      </xs:complexType>
      <xs:element name="Context" type="ContextType"/>
      <xs:complexType name="Context">
            <xs:sequence>
                  <xs:element name="id" type="xs:string"/>
                        <xs:element name="Constraint" type="ConstraintType"
                        minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

Annex C – The MCR Exchange Model (XML-schema)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://esdi-humboldt.eu/schemas/mediator"
xmlns:cs="http://esdi-humboldt.eu/schemas/context">
      <xs:element name="MediatorComplexRequest"
      type="MediatorComplexRequestType"/>
      <xs:import namespace=http://esdi-humboldt.eu/schemas/context
      schemaLocation="http://esdi-
            humboldt.eu/schemas/context/context.xsd"/>
            <xs:complexType name="MediatorComplexRequestType">
            <xs:sequence>
                  <xs:element name="id" type="xs:string"/>
                  <xs:element name="context" type="cs:ContextType"/>
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```