Title:

A5.2-D1 2.2 Mediator Service Component Specification

Editor(s)/Organisation(s):

Thorsten Reitz (Fraunhofer IGD)

Contributing Authors:

Thorsten Reitz (Fraunhofer IGD), Bernd Schneiders (LOGCMG), Marian de Vries (TUD)

References:

A1.8-D4 User Involvement Document - Second Version (555/3)
A5.1-D1 Framework specification Prototype (689/1)
A7.0-D1 Concept of Data Harmonisation Processes (707/1)
A5.2-D1 1.5 Workflow Service Component Specification (747/2)
A5.2-D1 1.4 Information Grounding Service Component Specification (748/2)
A5.2-D1 1.3 Context Service Component Specification (751/2)
A5.2-D1 1.0 Introduction and Specification Overview (762/1)
A5.2-D1 1.2 Model Repository Component Specification (765/1)
A5.2-D1 Framework Version 1.0 JavaDoc (766/1)

Status/Quality Assurance:

- ☑ Document Outline
- ☑ Content Draft
- ☑ Request for Comments
- ☐ Final
- ☐ Review WP Leader
- ☐ Review dependent WP Leaders (_____)
- ☐ Review ExB (_____)
- ☐ Review Associated Projects (_____)

Short Description:

This document details the specification of the Mediator Service Component, which handles orchestration of harmonisation processes and provides the results of these processes of several standard interfaces, such as WMS, WFS and WCS. It makes use of loosely-coupled, distributed processing capabilities and supports the discovery of harmonizable data sets, if the data sets to be transformed are not known in advance.

Keywords:

Framework specification, logical architecture, physical architecture, requirements, use cases.

History:

Version	Author(s)	Status	Comment
001-358	Thorsten Reitz	RFC	Initial version of the Mediator specification for Milestone 2 of the Framework specification

Table of contents

Figures

1 Introduction

1.1. Purpose of this document

This document describes a service component of the HUMBOLDT Framework, specifically the Mediator Service Component [2.2] (→ *and as such it is part of Deliverable A5.2-D2*). Beside outlining the purpose of the component and it's collaboration with other components, it contains information on the internal data models and message structures used by it. Data structures that it shares with other components are part of the HUMBOLDT Commons specification (→ *Document [2.1]*), which also represents Deliverable A5.3-D2.

The Mediator Service is a service component that executes transformation requests. It provides standard interfaces such as WCS, WPS, WFS and WMS for triggering these transformations and for returning results. It makes use of other HUMBOLDT components to get inputs to this process.

From the perspective of users who are working with geoinformation, the the Mediator Service is a very important part of the HUMBOLDT framework, and at the same time one that aims to be as invisible as possible. The component is designed as a proxy service, which interferes with normal usage of geoservices and data only in a minimal way, to provide users (specifically *end users of geoinformation,* → *User Involvement Strategy Document, Third Version*) with integrated services and harmonized geoinformation attuned to their needs, as defined via the Context Service (→ *Specification Document [2.4]*).

1.2. Abbreviations used in this document

This section summarizes the abbreviations used specifically for this service component. It does not repeat information found in the introduction and specification overview document [2.0].

Abbrev.	***Name***	***Definition***
AQM	Access Queue Manager	Module of the Mediator Service Component → *chapter 3.2.3*
ITF	Interface Translation Façade	Module of the Mediator Service Component → *chapter 3.2.1*
MCR	Mediator Complex Request	Container for the Constraints and Parameters collected from a User's request and possibly from the Context Service that is being used in many different places in the Mediator Service Component → *chapter 4.1*
RB	Request Broker	Module of the Mediator Service Component → *chapter 3.2.2*
TQM	Transformation Queue Manager	Module of the Mediator Service Component → *chapter 3.2.4*

1.3. Definitions valid in this document

This section gives important definitions valid for the context of this document. As in the section before, definitions available in the main document are not repeated.

Service Oriented Architecture (SOA)	Service Oriented Architecture is an architectural and organizational paradigm for organizing and utilizing distributed processing and storing capabilities that may be under the control of different ownership domains by defining loosely-coupled relationships between producers and consumers[1].

1.4. Standards used in this document

The Mediator Service Component makes use of some of the core standards in geoinformation for the provision of maps and other products, as well as raw data. Most notably, these are:

1.4.1 OGC Web Map Service and Styled Layer Descriptors (WMS with SE/SLD)

The WMS standard is used both as a Portrayal Service offered to the clients (WMS Interface Translation Façade implementation) and as a data provisioning service for background maps and similar purposes.

1.4.2 OGC Web Feature Service (WFS)

The Web Feature Service is, as the Web Map Service, used both for provisioning of data to the clients and for accessing data, i.e. the Mediator Service is both a WFS server and a WFS client.

1.4.3 OGC Geographic Markup Language (GML)

GML 3.2.1 parts are used for the core data model of the Mediator Service.

1.4.4 OGC Web Processing Service (WPS)

The WPS is used as a basic interface for the Transformer specification, which is used as the interface from the Mediator Service to processing capabilities.

1.4.5 ISO 19110:2005 Methodology for Feature Cataloguing

The structures presented in this standard are used to model conceptual schemas in the HUMBOLDT framework.

1.4.6 ISO 19115:2003

Many of the Metadata elements described in ISO 19115 are also used in the HUMBOLDT metadata management. Actually, the HUMBOLDT metadata is a profile of ISO 19115.

1 Based on the OASIS (Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org) and OMG (Object Management Group, http://www.omg.org) definitions of SOA.

2 Enterprise viewpoint

The Enterprise viewpoint describes the functional purpose of the component and it's integration with business processes. This chapter makes use of BPMN and of UML 2.0 Use Cases, as well as of the Volere Templating System for describing requirements.

2.1. Business process overview

The Mediator Service is part of one of the two main business processes that the HUMBOLDT Framework supports: It handles the actual execution of a transformation and provides the user with some support in specifying his product definition.

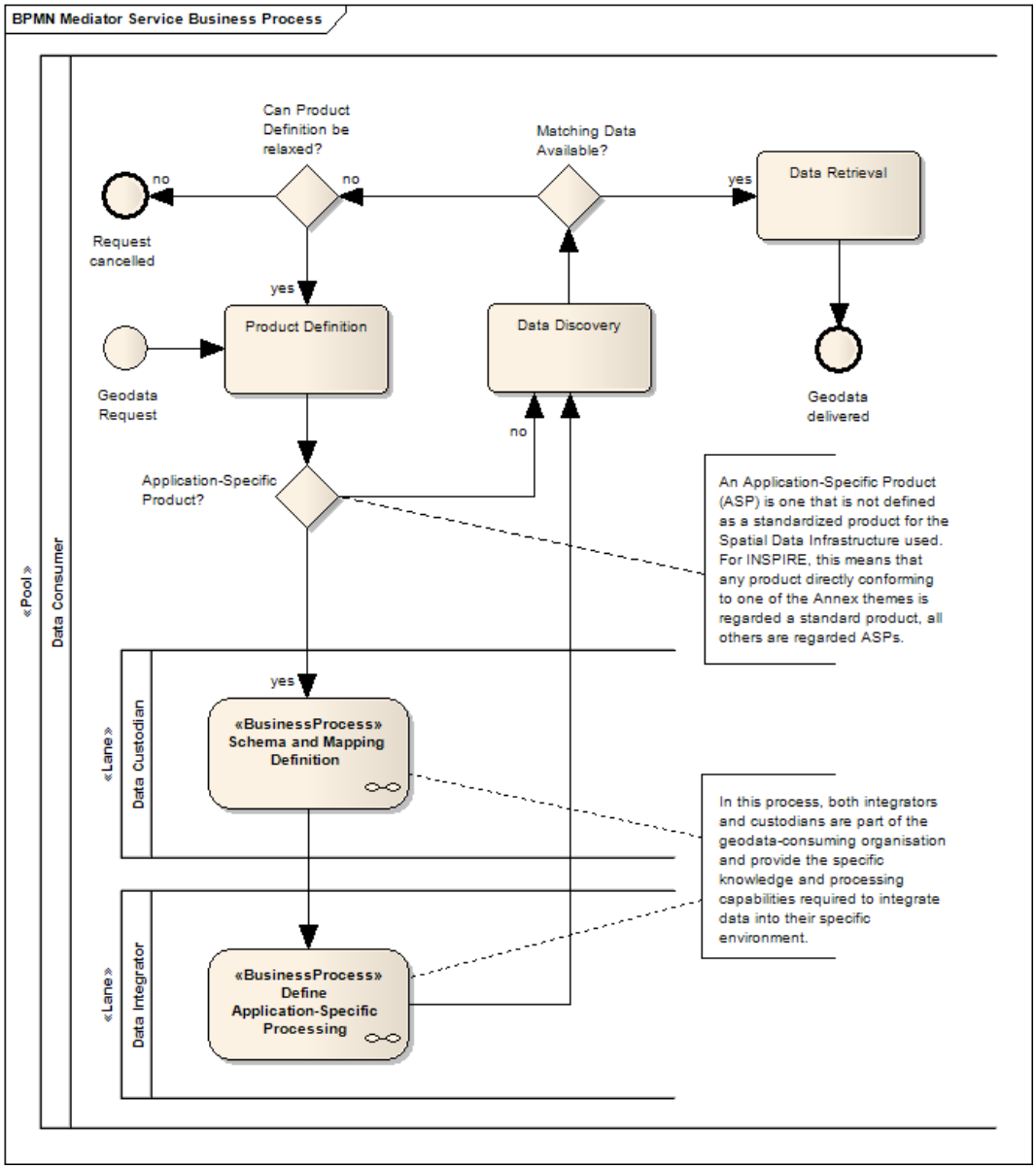


Figure 1: BPMN diagram showing the business activities that the Mediator Service supports

A typical usage scenario for the Mediator Service is the transformation of existing geodata in such a way that it is harmonised towards the INPIRE implementation rules. The Mediator Service can, in this scenario, be invoked with a well-known input data set (the non-harmonised one) and a product de-

scription that contains all the rules expressed by INSPIRE. Walking trough the business process depicted in Figure 1 for this harmonisation use case includes the following steps:

1. A *Request* to provide harmonised Geodata over a defined interface, in a specific format, conceptual schema, natural language and projection system is received. This request is usually not a technical request, but rather by mandate, since the organisation is affected by a national implementation of the INSPIRE directive.
2. This request is formalized and probably refined in a *Product Definition*.
3. Based on the Product Definition, the organization can now browse through their original, unharmonised data sets to *discover* which ones of these have a relevancy towards the request received in the first step. Often, this step will not be necessary since the organisation knows exactly which data sets need to be transformed to satisfy the request.
4. When matching data is discovered or predefined, the actual satisfaction of the request can take place and the data is delivered over an interface selected by the user (or at least prepared for *data transformation and retrieval*). This step means that the system will retrieve the source data to use and will apply a data-to-product-specific harmonisation processing workflow to it.

The Mediator Service Component also provides capabilities useful for many business processes in the HUMBOLDT scenarios. Here it also adds value by it's capability to orchestrate the retrieval of geoinformation and by it's capability to integrate and transform them to a client specified target description.

In the following, you'll find a concrete process description based on one of the scenarios as well as two abstracted process descriptions, showing the more general applicability of the component.

As an example how the Mediator Service is used, a Use Case from the HUMBOLDT scenario HS-ProtectedAreas is reflected: *HS04-06 Change Detection*. In this Use Case, the user (and End User of Geodata) needs to find geodata sets spanning a long period to perform an analysis of the changes that happened to a region in a protected area. These data sets are available from different providers over different interfaces, they use different formats and classifications, and the conceptual modelling applied was different. For this scenario, going through the Business Process depicted in Figure 1 looks as follows:

1. Again, the *Request* is received to prepare and deliver a certain geodata product, in this case, the time series of changes to the land cover of the protected area.
2. Now, the user first defines his requirements for this request by defining or selecting an applicable *Product Definition*, which is called a Context (→ *Context Service Component Specification [1.3]*) in the HUMBOLDT Framework. This Context contains information on the type of data sets required, on their required quality parameters (accuracy, completeness, ...) and on other properties.
3. Compared to the first process, we are now looking at an *Application-Specific Product* that is to be created: The goal is not to integrate geodata within a common infrastructure, but rather to use it within a specific environment. In this case, the environment is the protected areas management information system, towards which a fitting product has to be created from the data available in the common infrastructure. To enable this, the users have to go through the two business processes for *Schema and Mapping Definition* (→ *HUMBOLDT Model Editor Specification [2.3.1], HUMBOLDT Alignment Editor Specification [2.3.2]*) and possibly also for *defining Application-Specific Processing (→ Workflow Service Component Specification [2.5])*.

4. In the next step, which is *Data Discovery*, the *Schema and Mapping Definition*, and the *Product Definition* can now be used to see which of the available data sets are fit for the use within this use case, i.e. contain the right types and structures of data and also fulfil the quality requirements.
5. Finally, the system can now execute necessary transformations by orchestrating specific processing services, which handle the different aspects in which the geodata needs to be harmonised. With the addition of the *Application-Specific Processing*, a domain-specific process, in this case the analysis of the integrated/harmonised data for changes over time, can be added. This allows direct harmonisation towards an application target.

The user then selects the inputs he needs from this view and proceeds to start the retrieval of these data sets. H will receive these harmonized, i.e. transformed to a single, consistent physical, logical and conceptual schema, and can now make his analysis comparing the spatial attributes of two regions and how they evolve over time. To be able to make the analysis, various data sources containing the data from different spatial regions and different times have to be accessed and to be transformed in multiple aspects (such as Spatial Reference System, Geometric modelling, classification...) and to be provided to the user over a single interface, in this case, WMS and WFS.

Out of the business activities described above, the Mediator Service fully supports two and partially supports one. The ones that are supported are:

1. **Data Discovery:** The Mediator Service provides the Actors with a filtered view of the resources available in the known SDI. The filtering takes place according to the product definition. As an example, if a user has indicated that he is interested in a specific Feature Type from the Protected Sites Schema, the capabilities that are returned reflect availability of that dataset in the entire known SDI. Even Data Sets that can be transformed to fit the user's defined requirements are included. This capability reduces data discovery efforts on the side of END_USERS_OF_GEODATA.
2. **Data Transformation and Retrieval:** When the END_USERS_OF_GEODATA has decided precisely which data to use (i.e. he has completed and refined his Product Definition), the Mediator Service handles the complexity of doing all necessary data retrieval, fusion, transformations and encoding. In the end, the specified product is delivered according to the definition given by the END_USERS_OF_GEODATA or used by the END_USERS_OF_GEOINFORMATION.
3. **Product Definition:** This is a partially supported activity that is mainly done using the Context Service, but the refinement of the product definition based on actually available data sets is done because of the Mediator Service's provision of optimized Discovery mechanisms (see point 1 of this list).

The business process also shows that if an END_USER_OF_GEODATA defines a product that is not a predefined product standardized in the SDI being used, two ancillary business processes might have to be invoked. Following the examples above, a standardized product in the INSPIRE-based European SDI could be a protected sites dataset (specified through the Data Specification for the Annex I theme), whereas an application-specific profile of that data set is considered an ASP (Application-Specific Product). For this latter case, the definition of the target profile, the definition of transformation rules from the harmonised schema to the ASP and the provision of specific processing capabilities will be necessary.

END_USERS_OF_GEOINFORMATION are only expected to use predefined Product Definitions for which all schemas, mappings and processing services have already been set up.

It should be noted that the Mediation process can be done both as an off-line migration (large subsets/ expensive processing), or as an on-line (aka on-the-fly) harmonization. The off-line migration is recom-

mended if the requests would usually contain large subsets of data or involve expensive processing, whereas the on-line harmonization is especially recommendable when high flexibility is required and when small subsets of a large data set are required that only need rather cheap processing. Finally, the two modes can actually be combined by making use of caching technology for temporary storage of often-used, expensive to compute mediation results. In the example above, the researcher might use on-line harmonization during his work, but for the provision of the portrayed product, it would of course make sense to prepare it off-line at least partially, especially if the base data is not subject to change for the context at hand.

As a side effect of the discovery activities that are supported, the Mediator Service can also be used to get a task-oriented, domain-specific view on the known resources in an SDI by using a Product Definition to filter available resources. Thus, the available services in the SDI are filtered down to the used conceptual schema of the application domain in question, allowing a user to discover information matching the required concepts directly instead of data set oriented services and doing so more efficiently, since all available metadata is matched against the context. This process has the following major steps:

1. *Retrieval of target requirements:* As in the geodata retrieval process, these define the goal of the user and are usually defined in a so-called Context document.
2. *Retrieval of Conceptual Schema:* A Conceptual Schema represents a formalized way of expressing the concepts and terms of a certain application domain. This is retrieved from an external service (→ *Model Repository Service Component [1.2]*)
3. *Retrieve Groundings for Schema Elements:* This step now means doing an automated discovery of the services principally available per conceptual schema element. You can think of this as identifying the services capable of providing a certain layer, such as land usage, only that no service-specific layer name is used, but the more abstract notion of a Feature Type (concept) that can have many terms representing it. For performing this step, the Information Grounding Service is invoked (→ *Information Grounding Service Specification [1.4]*).
4. *Filter down on additional requirements:* Finally, this last check narrows down the view in aspects that cannot be handled by normal catalogues by retrieving the grounding services' capabilities and checking those against the target requirements. The result is a view of the SDI, expressed as a capabilities document, tailored towards a certain application domain.

Compared to other existing solutions on the market, the HUMBOLDT Mediator Service offers the technical USPs of being able to orchestrate OGC processing and download services, of being able to act as an invisible proxy for WMS, WPS, WCS and WFS interfaces and of providing a new method of service and data discovery. It's value proposition in terms of qualitative and organizational advantages includes:

- less effort in data discovery and integration through sophisticated automation
- increased quality of data integration
- better accountability and traceability of the integration process
- good integration with existing infrastructure due to usage of common interfaces

Also in the focus of this specification was a fulfilment of the critical success factors for information related business processes, as stated in A2.2-D1:

- Fast delivery of the service
  - through online/offline/mixed approach
- The information shall be of high quality (correct, relevant and up to date)

- best available information is identified and retrieved on-line

- The service shall be flexible (customizable to user's needs)
  - via Context documents and Task/Workflow definitions, very wide ranges of user needs can be covered
- The service shall be provided at low cost
  - does not require huge storage, can make good use of existing infrastructure
- High convenience
  - the Mediator Service is designed to be almost invisible by offering established interfaces
- Reliable, always provide a consistent result
  - documentation of all processes

2.2. Application in the Protected Areas Scenario

This component plays a role for all scenarios,since it orchestrates the execution of transformations necessary to harmonise data. Because of the special status of the Protected Areas Scenario as a Fast Track Scenario, this section highlights the usage of the Mediator Service in the fast Track Scenario. For general information on the FTS PA, please refer to the PA documentation.

After Mario has created the context representing the footpaths data set that should be used for tourism valorisation, he can now use his standard GIS to continue work. In his GIS, he selects the WFS interface from which to import data. In the dialogue that pops up, he enters the URL of the Mediator Service as given to him by the Context Service.

Next, the capabilities are retrieved from that service. The response that is now being displayed contains all the items that are part of the harmonised application schema for the scenario that was created by Mario and his colleagues a while ago. He can select any of the FeatureTypes in that schema, and additionally one that he added for his specific tasks. In this case, he picks one of his special types, the type "Sustainable touristic footpaths". In the background, information on that Feature Type is now being retrieved. This information, together with metadata such as the area for which this type is available, is now being displayed. Mario can now set any filters he wants, such as only returning footpaths with an accumulated ascent of less than 1000 meters, by using the filter encoding UI.

Figure 2: The Network of paths after the processing in the Mediator Service

After doing so, he submits his initial query and after a short while, he receives the footpaths from the four regions. The footpaths have been fused to a single dataset, with which Mario can now continue to work.

The full process behind this user experience involves several additional steps, which are described in Chapter 3 of the Specification Introduction and Overview Document [2.0].

2.3. Actors in this component

As shown in the scenarios analysis, the most important actors for this component are the **End Users of Spatial Information and of Geodata** for information discovery and transformation. For the **System Integrators**, this component is also of some importance, but mainly from an administrative perspective. The **Data Custodians** use the component in mainly the same way as the end users, to verify settings they have created using other components.

The actors for this component are consequently as follows:

Actor Name	***Actor Description***
MS_END_USER	For the purpose of this component, both End Users of Geoinformation and End Users of Geodata can be summarized into one actor. This actor uses the Mediator Service just as it would use any other geodata service, usually integrated into a wider application such as a scenario-specific portal or desktop application.
SYSTEM_INTEGRATOR	This is the standard system integrator actor from the introduction and specification overview document, chapter 4.2.1.
DATA_CUSTODIAN	This is the standard data custodian actor from the introduction and specification overview document, chapter 4.2.1.

No additional component-specific actors are introduced.

2.4. Mediator Service Component Use Cases

This section summarizes the Use Cases which this component has to fulfil. It should be noted that most of the Use Cases below have a common set of preconditions, as also described in the business process in the introduction and specification overview document in chapter 4.2. These preconditions are:

- The HUMBOLDT (horizontal) infrastructure has been set up, including the Context Service, Model Repository, the Workflow Service and the Information Grounding Service.
- Target and source conceptual schemas have been described by the DATA_CUSTODIANs and have been mapped, so that a schema transformation can be executed based on this mapping. Schemas and mappings have been published to the Model Repository. This can be considered a type of preprocessing.
- A Product Definition in the form of one or more Context documents has been defined for the metadata and geodata retrieval and has been stored in the Context Service.
- For those cases where an Application-Specific Product is requested, a basic workflow has been created by the DATA_INTEGRATOR, including the registration of all necessary processing capabilities (transformers) with the Workflow service.

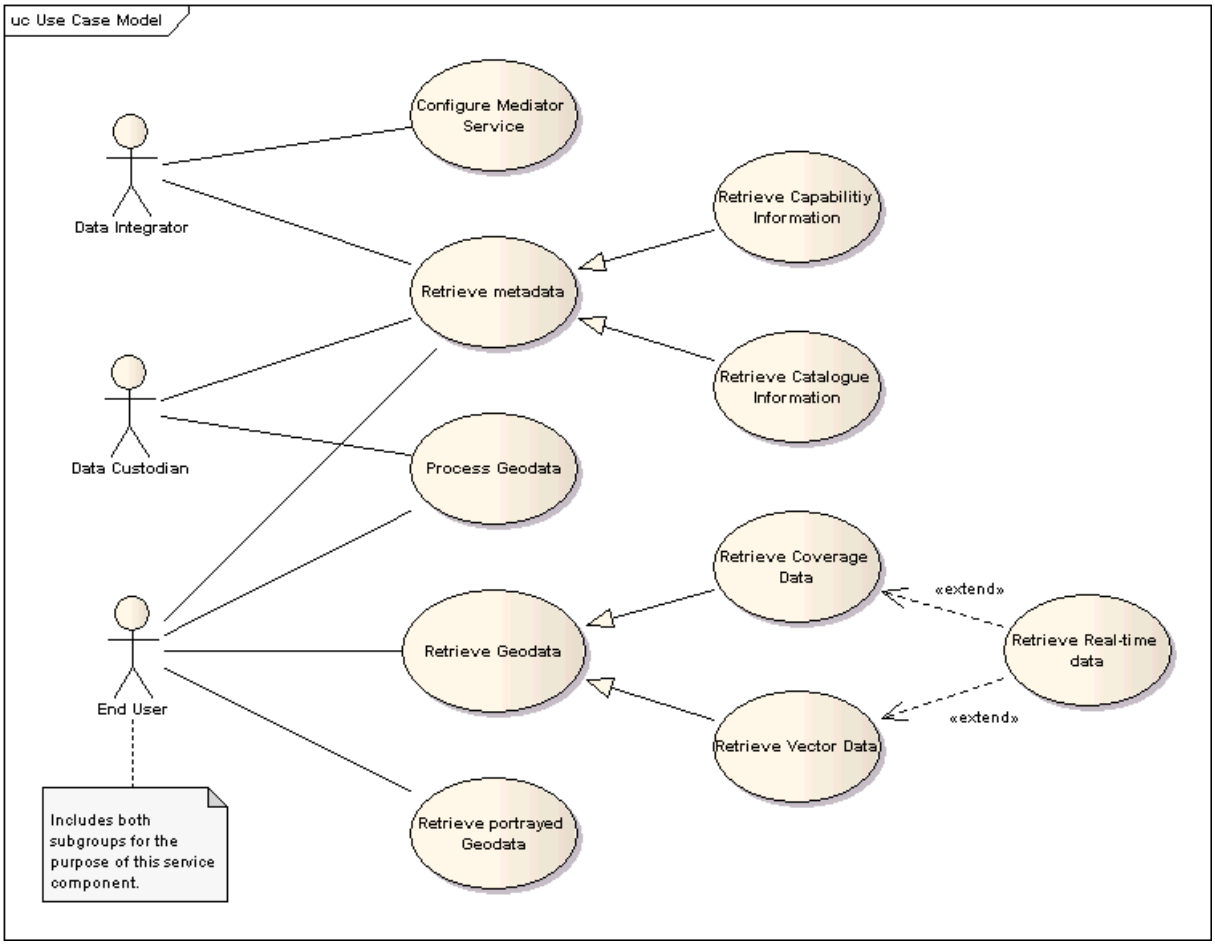


Figure 3: UML Use Case diagram with the major Use Cases for the Mediator Service Component

The main group of Use Cases relates to the End User actor, who does not want to be bothered with integration issues and also wants to have simple means of data discovery. For that purpose, four Use cases have been described:

- *MS01: Retrieve portrayed Geodata:* provide the client with a finished product, such as a thematic map, based on heterogeneous and possibly distributed geodata
- *MS02: Retrieve harmonized Geodata for further local processing:* This involves combining data from various data sources that are heterogeneous in one or more aspects and providing them in such a way that further processing, such as in a Desktop GIS, can be applied. This Use Case includes several variants dealing with coverages or vector data, and also with real-time streams.
- *MS03: Process Geodata:* In this Use Case, the user is requesting a processing result based on data provided by him to the service.
- *MS04: Retrieve Metadata:* This Use Case is used mainly for discovery of services. Since the MS provides virtual services, the concrete capabilities depend on the services supported by a concrete instance of the Mediator Service. This Use Case includes two specializations, used to retrieve service capabilities (such as retrieved via a WMS `getCapabiltities` operation) and catalogue data (such as via a CSW `getRecords` operation).
- *MS05: Configure Mediator Service:* This is a Use Case invisible to the other user groups. It allows system / data integrators to define the downstream services used, the available interfaces, internal processing models and similar things.

The following sections detail these main Use Cases.

2.4.1 MS01 – Retrieve portrayed Geodata

Version: 2.0, 26. Jan. 2009

1. Description

An End User wants to retrieve a finished product based on geospatial data, such as a rendered raster or vector map. On the client side, this is usually done via an interface such as the Standard GIS Client, the Thin Client Component or the Rich Client Component. These interfaces reflect common GIS interaction strategies, such as using Layers to identify the themes of the map, or have a simplified just-navigation interface for maps for a single purpose.

This Use Case is abstracted, among others, from the following scenario Use Cases:

- HS04-02: Tourism valorisation in a protected area
- HS04-03: Impacts of infrastructure on protected areas
- HS02-02: Providing data of territorial planning bases and territorial planning documentation to a planner of the territorial plan
- HS03-03: Locality/object searching from a map, finding GPS reference

2. Actors

MS_END_USER	An end user accessing the MS_SYSTEM via one of it's exposed interfaces, using a compatible client.
MS_SYSTEM	A deployed, configured instance of the Mediator Service Component offering the portrayal interface required by the client type used by MS_END_USER.

3. Initial Conditions

The Client that MS_END_USER uses has been configured to use the MS_SYSTEM's application URL as hostname, including a context identifier, either specified by MS_END_USER or an unbound task context (Please refer to Context Service Component Specification [1.5]). This context is required for determining processing and harmonization requirements for the task at hand.

4. Final Results

The User's Client is provided with a finished portrayed product, such as a vector or raster map.

5. Processing

Main Process

1. MS_END_USER works in his client application and selects the thematic groups he wants to have included in his map product.
    a. MS_SYSTEM confirms availability of information by discovering required services.
    b. MS_SYSTEM confirms to the MS_END_USER that it can fulfil the request or asks for other constraints.
    c. MS_SYSTEM queries discovered services and processes results according to constraints.
    d. MS_SYSTEM sends portrayed product to MS_END_USER.
2. MS_END_USER navigates interactively through the map.
    a. MS_SYSTEM provides updates to the map product as necessary.

Alternative Processes

AP01: An unbound task context is used.

Instead of step 1 / a. to b.:

1. MS_END_USER works in a portal and comes to the page where the portrayed map product has been integrated.
    a. MS_SYSTEM uses predefined information for quick service discovery, processing workflow set-up and execution.

Exceptional situations

1. ES01: Context ID not available or no Context found for ID: In this case, the Mediator Service Component will fall back to its DefaultContext.
2. ES02: No services are available to match MS_END_USER's constraints: MS_SYSTEM will notify the user about the reasons for not finding a result and will try to suggest an alternative, such as a reduced resolution.

6. Processed data

This depends mainly on the content of the user's requirements, expressed by the usage of his client's interface and by the context contents. In a typical Use Case from one of the scenarios, the processed data will look like this:

Input:

– Raster Imagery (satellite images, orthophotos) retrieved and used for background

– Thematic vector data retrieved from heterogeneous sources (each covering different spatial areas, times or themes (`FeatureTypes`)) harmonized to the user's constraints

– Labels and metadata, translated to the user's constraints regarding conceptual schema and natural language

Output:

– Single Raster image, with the thematic vector data and labels applied to it, using the portrayal information from the context or request

7. Generated Data

The output is sent to the client. No data is persisted from the side of the Mediator Service Component.

8. Rules of Processing

No specific rules of processing have been defined at the moment.

2.4.2 MS02 – Retrieve harmonized Geodata for local processing

Version: 2.0, 26. Jan. 2009

1. Description

An End User of Geodata (MS_END_USER) wants to retrieve geospatial data, such as a set of layers, for further processing in his client. This is a Use Case found quite often in the HUMBOLDT scenarios, such as in HS-ProtectedAreas and HS-Water.

On the client side, this is usually done via an interface such as the Standard GIS Client or the Rich Client Component. The selection is managed via common interaction means typical to GIS; such as navigating through a base map and selecting themes from a layer list or from an information model view.

This Use Case has two specializations, which are to the user mostly identical, but internally require different processing:

– MS02.a Retrieval of Coverage geodata

– MS02.b Retrieval of Vector geodata

This Use Case is abstracted from the following scenario Use Cases:

– HS02-01: Providing of preliminary information – territorial planning information

– HS02-04: Providing data of structural ecological network parts – SSES (spatial system of ecological stability) to applicant / expert

– HS02-05: Providing data of structural ecological network parts – SSES (spatial system of ecological stability) to applicant / nonexpert

– HS03-02: Locality/object searching from GPS reference

2. Actors

MS_END_USER	An end user accessing the MS_SYSTEM via one of it's exposed interfaces, using a compatible client.

MS_SYSTEM	A deployed, configured instance of the Mediator Service Component offering the portrayal interface required by the client type used by MS_END_USER.

3. Initial Conditions

The conditions are similar to the ones in MS01 - the Client that MS_END_USER has been configured to use the MS_SYSTEM's application URL as hostname, including a context identifier,. The usage of a task context is not part of the Use Case. This context is required for determining processing and harmonization requirements for the task at hand.

4. Final Results

The User's Client is provided with the geodata he requested, processed and transformed according to the constraints specified. The returned data can be coverages (MS02.a) or vector data (MS02.b).

5. Processing

Main Process

For both variants to this Use Case (MS02.a and MS02.b), the process is identical.

1. MS_END_USER works in his client application and configures the parameters that can be configured via the interface that the client supports, such as thematic groups, spatial and temporal bounds, resolution and metadata.
   a. MS_SYSTEM confirms availability of information by discovering required services.
   b. MS_SYSTEM confirms to the MS_END_USER that it can fulfil the request or asks for other constraints.
   c. MS_SYSTEM queries matching services and processes results according to constraints.
   d. MS_SYSTEM sends processed geodata to MS_END_USER.
2. MS_END_USER works with the geodata set.

Alternative Processes

There are no alternative processes currently described.

Exceptional situations

1. ES01: Context ID not available or no Context found for ID: In this case, the Mediator Service Component will fall back to it's DefaultContext.
2. ES02: No services are available to match MS_END_USER's constraints: MS_SYSTEM will notify the user about the reasons for not finding a result and will try to suggest an alternative, such as a reduced resolution.

6. Processed data

As in MS01, this depends mainly on the content of the user's requirements, expressed by the usage of his client's interface and by the context contents. In a typical Use Case from one of the scenarios, the processed data might look like this:

Input:

– Thematic vector data retrieved from heterogeneous sources (each covering different spatial areas, times or themes (`FeatureTypes`)) harmonized to the user's constraints

Output:

– Single data set of thematic vector data with labels and metadata transformed using the constraints from the context or request.

7. Generated Data

In Use Case MS02.a, a harmonized coverage is created and sent to the client. In Use Case MS02.b, a vector data set is created and sent to the client. In either case, no data is persisted on the side of the Mediator Service Component.

8. Rules of Processing

No specific rules of processing have been defined at the moment.

2.4.3 MS03 – Process Geodata

Version: 2.0, 23. Jan. 2009

1. Description

An End User of Geodata (MS_END_USER) wants to create a report or to otherwise process geodata in a decision-supporting way. For this purpose, he provides as input besides his (task) context geoinformation that he locally prepared. In this Use Case, the actor DATA_CUSTODIAN can also become active, using the same facilities to make harmonized data available. For the purpose of this Use Case, both are considered to be MS_END_USERs.

This Use case has two specializations, which are to the system mostly identical, but result in a different behaviour towards the user:

– MS03.a Processing of geodata and returning of result data sets

– MS03.b Processing and persisting the result data set

This Use Case is abstracted from the following scenario Use Cases:

– HS02-06/07: Equalization data by mains administrator to accredited authority

– HS03-04: Mapping – inserting new geographical entities

– HS03-05: Mapping – geographical entities editing

– HS04-01: Management of a protected area

2. Actors

MS_END_USER	An end user or data custodian accessing the MS_SYSTEM via one of it's exposed interfaces, specifically an interface that accepts geodata, such as WFS-T or WPS, using a compatible client.
MS_SYSTEM	A deployed, configured instance of the Mediator Service Component offering the portrayal interface required by the client type used by MS_END_USER.

3. Initial Conditions

The conditions are similar to the ones in MS01 - the Client that MS_END_USER has been configured to use the MS_SYSTEM's application URL as hostname, including a context identifier. This context is required for determining processing and harmonization requirements for the task at hand.

4. Final Results

The User's Client is provided with the processing result he requested, processed and transformed according to the constraints specified. The returned data can be coverages (MS03.a) or vector data (MS03.b).

5. Processing

Main Process

For both variants to this Use Case (MS03.a and MS03.b), the process is identical.

1. MS_END_USER loads geoinformation from local sources, such as a sensor or a file, or creates additional geoinformation in his client application.
2. MS_END_USER configures the parameters that can be configured via the interface that the client supports, such as thematic groups, spatial and temporal bounds, resolution and metadata. In addition, he adds his newly-created geoinformation to the query.
    a. MS_SYSTEM confirms availability of information by discovering required services and verifies the input.
    b. MS_SYSTEM confirms to the MS_END_USER that it can fulfil the request or asks for other constraints.
    c. MS_SYSTEM queries matching services and processes both input and results according to constraints.
    d. MS_SYSTEM sends processed geodata to MS_END_USER.
3. MS_END_USER works with the result geodata set.

Alternative Processes

AP01: A user saves data to the system, which might need processing, but does not expect data back.

Exceptional situations

1. ES01: Context ID not available or no Context found for ID: In this case, MS_SYSTEM will fall back to it's DefaultContext.
2. ES02: No services are available to match MS_END_USER's constraints: MS_SYSTEM will notify the user about the reasons for not finding a result and will try to suggest an alternative, such as a reduced resolution.
3. ES03: Interpreting the provided data failed: MS_SYSTEM informs the user of the unrecoverable errors in the data and suggests corrections, if applicable.

6. Processed data

Again, this depends mainly on the content of the user's requirements, expressed by the usage of his client's interface and by the context contents. However, an additional element is added to the input side in a typical Use Case from one of the scenarios:

Input:

- Thematic vector data retrieved from heterogeneous sources (each covering different spatial areas, times or themes (feature types)) harmonized to the user's constraints
- Thematic vector data provided by the Client

Output:

- Single data set of thematic vector data with labels and metadata transformed using the constraints from the context or request.

7. Generated Data

In Use Case MS03.a, a response data set, either a coverage or a vector set, is created and sent to the client. In Use Case MS03.b (saving data), a confirmation is returned after the transaction is closed.

8. Rules of Processing

No specific rules of processing have been defined at the moment.

2.4.4 MS04 – Retrieve Metadata

Version: 1.0, 14. Jan. 2008

1. Description

This Use Case is often the first one to be executed when working with the Mediator Service.

MS_END_USER, a DATA_CUSTODIAN or a DATA_INTEGRATOR wants to discover what kind of services and information are available through the MS_SYSTEM he is using. Again, these users are working from within a standard or custom rich client based on the standard interfaces implemented in MS_SYSTEM and are using a defined context. These services can on the one hand provide catalogue-like functionality, or on the other hand express what capabilities are offered via one of the standard interfaces.

The responses to this are always built on the base of the user's constraints, especially his conceptual schema and spatiotemporal constraints as defined in the currently active context.

The Use Case has two specializations:

MS04.a: Retrieval of Capabilities information

MS04.b: Retrieval of Catalogue data

2. Actors

MS_END_USER	An end user, data custodian or data integrator accessing the MD_SYSTEM via one of it's exposed interfaces that can provide metadata, such as the GetCapabilities documents provided by all OGC services or operations provided by the OGC CWS specification.

MS_SYSTEM	A deployed, configured instance of the Mediator Service Component offering the portrayal interface required by the client type used by MS_END_USER.

3. Initial Conditions

The conditions are similar to the ones in MS01 - the Client that MS_END_USER has been configured to use the MS_SYSTEM's application URL as hostname, including a context identifier. This context is required for determining processing and harmonization requirements for the task at hand. Having no context would mean an infinite number of possibilities in the capabilities and catalogue information.

4. Final Results

The User's Client is provided with the metadata he requested, processed and transformed according to the constraints specified, such as natural language and conceptual schema. In both variants of the Use Case, the data is basically a list of possible service usages.

5. Processing

Main Process

The process is identical for both variants of the Use case.

1. MS_END_USER selects the service type he wants to get the capabilities of in his client application.
   a. MS_SYSTEM discovers available services and verifies the input.
   b. MS_SYSTEM filters the results down to those matching user-specified constraints.
   c. MS_SYSTEM processes results to match MS_END_USER's additional constraints, such as language and conceptual schema.
   d. MS_SYSTEM sends processed geodata to MS_END_USER.
2. MS_END_USER works with the result metadata set.

Alternative Processes

None.

Exceptional situations

1. ES01: Context ID not available or no Context found for ID: In this case, MS_SYSTEM will fall back to it's DefaultContext.
2. ES02: No services are available to match MS_END_USER's constraints: MS_SYSTEM will notify the user about the reasons for not finding a result and will try to suggest an alternative, such as a reduced resolution.

6. Processed data

In this case, virtual service descriptions are generated by creating the intersection between known actual services and the constraints expressed by the user. This can either be done in a single-service-centric way, which in it's capabilities description might actually accumulate a lot of services, or in a multiple-service-centric way, which lists services in catalogue style for each of the thematic constraints expressed.

7. Generated Data

In Use Case MS04, a response metadata set, is created and sent to the client. No data is persisted on the side of MS_SYSTEM.

8. Rules of Processing

No specific rules of processing have been defined at the moment.

2.4.5 MS05 – Configure Mediator Service

Version: 1.0, 14. Jan. 2008

1. Description

A DATA_INTEGRATOR wants to either monitor MS_SYSTEM instance status, or wants to change the internal state. This involves registering downstream, HUMBOLDT services that can be used, configuring which service interface the instance exposes, which internal processing setting it uses, such as conceptual schema and natural language, and many other parameters. This Use case is of no importance for the other user groups.

This Use Case is not directly abstracted from any Use cases mentioned in the HUMBOLDT scenarios so far.

2. Actors

DATA_INTEGRATOR	A data integrator accessing MD_SYSTEM via a specific interface for the administration and management of a MS_SYSTEM instance, with the aim of controlling instance status or configuring internal state.
MS_SYSTEM	A deployed, configured instance of the Mediator Service Component offering the portrayal interface required by the client type used by MS_END_USER.

3. Initial Conditions

To use the configuration capabilities, DATA_INTEGRATOR needs to log on with his account with administration privileges to one of the clients supporting the management and configuration interfaces.

4. Final Results

DATA_INTEGRATOR completes the configuration and leaves behind a working, correctly-configured MS_SYSTEM instance.

5. Processing

Main Process

1. DATA_INTEGRATOR uses the client of his choice to access the configuration of the MS_SYSTEM.
   a. MS_SYSTEM displays the configuration or part thereof requested by DATA_INTEGRATOR.
2. DATA_INTEGRATOR edits the configuration and submits the changes to MS_SYSTEM.

a. MS_SYSTEM processes the edits and checks whether they are consistent and valid.

b. MS_SYSTEM confirms setting the new values or returns an error message, stating to DATA_INTEGRATOR which value was not allowable and what the allowed value range is for that parameter.

3. DATA_INTEGRATOR repeats this process until all settings are as intended.

Alternative Processes

None.

Exceptional situations

1. ES01: The configuration is in an invalid or inconsistent state: The DATA_INTEGRATOR is informed of this via a push medium.

6. Processed data

The configuration data for the system is being processed here. This data consists of set-up and composition information, i.e. information on which Workflow Service instance, which Model Repository instance and so on to use, as well as configuration items like information about which users can use which interfaces and services. Finally, the internal state and the processing profile can be set.

7. Generated Data

No additional data is generated on the side of MS_SYSTEM.

8. Rules of Processing

No specific rules of processing have been defined at the moment.

2.5. Requirements

This chapter contains both the functional requirements derived from the Scenarios/Use Cases and the non-functional requirements from the HUMBOLDT scenarios and other sources.

2.5.1 Functional Requirements

These functional requirements have been exported from the Volere Requirements Management tool.

1. **I/O Requirements**
   - **Offered Interface Requirements**
     - The system must offer all mandatory and optional OGC Web Map Service operations to clients invoking it.
       - The system should accept client-provided SLD and SE and render output accordingly.
     - The system must offer all mandatory and optional OGC Web Feature Service operations to clients invoking it.
       - The system must support Filter Encoding for WFS queries.
     - The system should offer all mandatory and optional OGC Web Coverage Service operations to clients invoking it.
     - The system should offer all mandatory and optional OGC Web Processing Service operations to clients invoking it.
   - **Required Interface Requirements**
     - The system must be able to invoke OGC Web Processing Services.
       - The system must be able to invoke WPS execute operations with all parameters required, as indicated by the workflow definition received from the Workflow Service.
     - The system should be able to invoke WSDL- and SOAP-based Web Services.
     - The system must be able to load data from a Web Feature Service.
       - If a remote WPS is invoked, the system must pass the reference URL indicating the correct `WFS:GetFeature` operation as part of the `WPS:ExecuteRequest`.
     - The system should be able to load data from a Web Map Service.
     - The system should be able to load data from a Web Coverage Service.
     - For all required interfaces, the system should check the availability of the services it wants to invoke before starting the execution of the process chain.
     - For all required interfaces, the system must handle an exception or non-availability of the invoked service in such a way that the client invoking the system gets informed.

2. **Functional Requirements**
   - The SYSTEM must be able to create a request from the combination of the context and the request received via the interface exposed to the client.
     - In the case of a GetCapabilities operation of any interface that offers such an operation, the SYSTEM must provide a response that contains the capabilities of the known infrastructure, filtered by the constraints provided by the used context.

        - The Capabilities that are returned by the SYSTEM must be based on the services known to the Information Grounding Service.
        - The Capabilities of these services must be combined and filtered by the SYSTEM according to the constraints defined in the context valid for the GetCapabilities request.
        - To determine which services fulfill the constraints defined in the context valid for the GetCapabilities request, the SYSTEM must also allow FeatureTypes {B} for which a transformation description exists from {B} to the FeatureType(s) given as input {A}.
            - To determine which FeatureTypes {B} can be transformed into the requested FeatureTypes {A}, the SYSTEM must invoke the Model Repository Service.
    - In the case of any type of "Get" operation, such as GetMap or GetFeature, the SYSTEM must assemble a response according to the format expected by the client.
- The SYSTEM must be able to execute a workflow which consists of the following elements: Calls to grounding download services such as WCS and WFS and calls to grounding transformation services, such as WPS.
- If the SYSTEM receives a query for more than one FeatureType, the SYSTEM must request and execute a workflow for each one individually and must assemble the outcomes of each FeatureTypes's request in such a way that it matches the CLIENT's request.

3. Interaction with other HUMBOLDT components requirements

- The SYSTEM must be able to send a request (Mediator Complex Request, please refer to HUMBOLDT Commons) to the Workflow Service to retrieve an executable workflow.
    - Such a workflow must contain information on the services to use as data sources and transformers and must contain all information to create complete requests to dispatch to those services.
    - In the case that an executable workflow cannot be returned, the SYSTEM must send an Exception to the CLIENT, explaining which Constraint could not be fulfilled.
- The SYSTEM must be able to retrieve a specific context from the Context Service.

4. Technical Requirements

- The SYSTEM should cache executable workflows.
    - The SYSTEM should use the capabilities information from services that it invokes during workflow execution to build information on when a workflow can be re-used (i.e. to determine whether a request is a cache-hit).
        - The information that the SYSTEM stores alongside with the executable workflow to determine cache-hits should include the spatial extent that the services chained in the service can provide.

3 Service and Computational viewpoint

This chapter details the services and APIs that this component provides, as well as the internal logical architecture and processes used.

3.1. Overview of the Mediator Service Component

This component contains many of the HUMBOLDT framework's internal processes and can be seen as the Façade[2] and Controller[3] of a HUMBOLDT installation for End User queries. It is also of relevance to Data Custodians and System Integrators, as detailed in the Enterprise Viewpoint description of the component. Logically, the Mediator Service Component consists of the following modules:

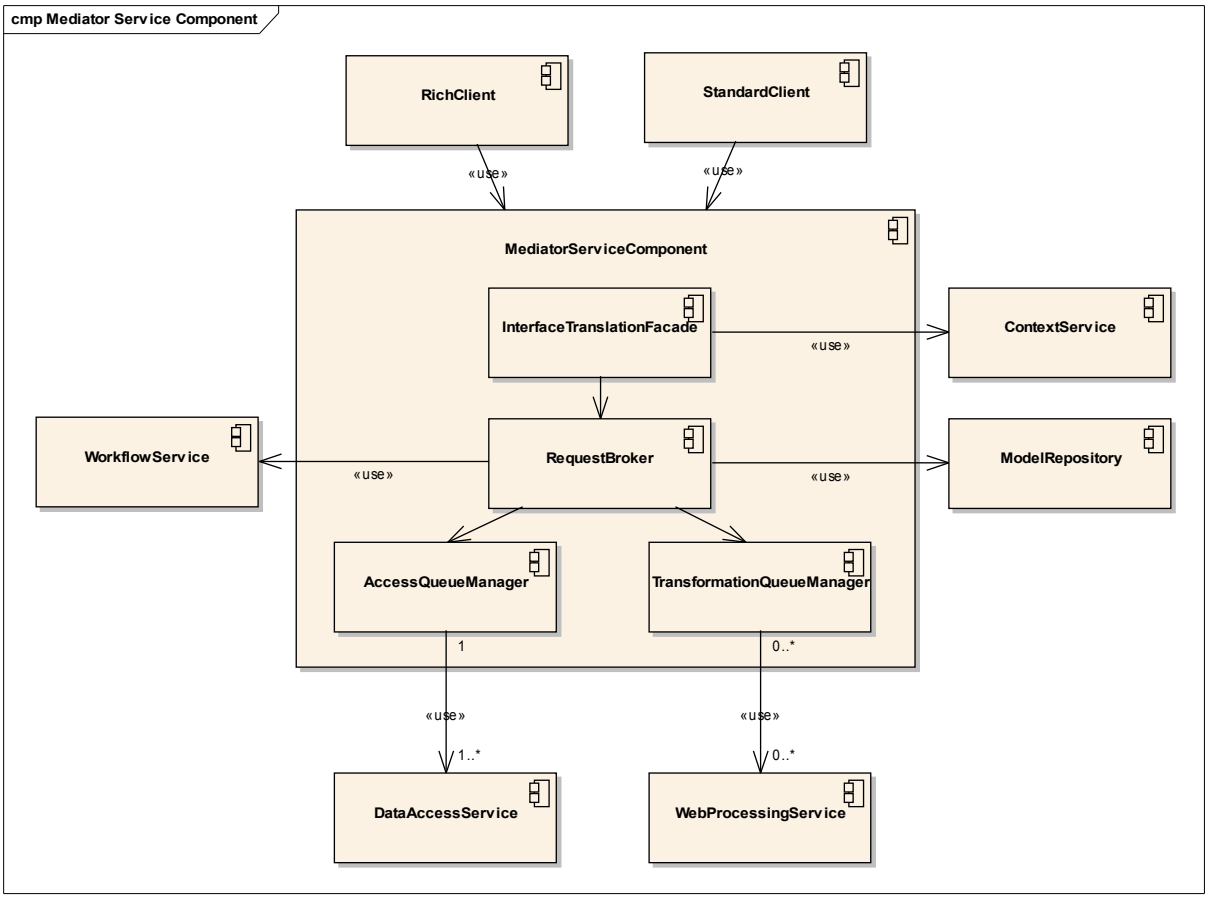


Figure 4: Component Diagram of the Mediator Service Component, with dependencies to other service components of the HUMBOLDT Framework

2 The *Façade Pattern* is one of the most common architectural patterns in software engineering. It is a structure that provides a simplified interface to a larger body of code, in this case, the diverse HUMBOLDT services. In terms of the *Model View Controller (MVC)* architectural pattern, it represents the *View*, i.e. a representation of the internal model that is suitable for client interaction. For more information, refer to Gamma et al. (1998), *Design Patterns: Elements of Reusable Object-Oriented Software* (ISBN 0-201-63361-2) and to Fowler, Martin (2006), GUI Architectures, On-line resource: http://www.martinfowler.com/eaaDev/uiArchs.html.

3 The *Controller* in the *Model View Controller (MVC)* architectural pattern can be understood as the steering logic of a system, invoking subcomponents as necessary and may invoke changes on the model. For more information on the MVC pattern, please refer to Fowler, Martin (2006), GUI Architectures, On-line resource: http://www.martinfowler.com/eaaDev/uiArchs.html.

- *Request Broker (RB):* This component is the main logical controller of the mediator. It accepts a request and invokes the Transformation Queue and the Access Queue to satisfy the particular requirements imposed by that request. It also invokes the Workflow Service to retrieve an executable Workflow for processing a given request.
- *Access Queue Manager (AQM):* This component collects access requests, i.e. requests for data that come from the Request Broker and have been determined based on the prerequisites of the workflow provided by the Workflow Service and executed in the Transformation Queue Manager.
- *Transformation Queue Manager (TQM):* In this component, both request and response elements are transformed on the basis of the workflow established for a specific request. The Transformation Queue manages the execution of the individual transformations (which can be local algorithms or remote processing services) and notifies the Request Broker when it needs additional data or is finished.

The main process inside the Mediator Service Component, using these components and invoking several of the other service components available in the HUMBOLDT framework, can be summarized as follows (please also refer to Figure 5, p. 28):

1. A client sends a request, such as a Web Map Service `GetMap` request, to the Mediator Service Component, including a context identifier registered beforehand with the Context Service or a complete `MediatorComplexRequest` (in the case of a Rich Client using the Rich Query Service).
2. The Interface Translation Façade accepts the request, decodes it and constructs an enriched request object, the `MediatorComplexRequest`, from it. It also uses the identification information in the URL to retrieve the User's harmonization preferences from the Context Service.
3. The `MediatorComplexRequest` is then delegated to the `RequestBroker`, effectively the system controller. Before continuing the execution of the request, the `RequestBroker` first asks the `ModelRepository` whether the used Information Model is known and whether it is identical to the information model defined as the internal model for this instance. If the latter is not true, the (thematic) query constraints are conceptually translated to the internal information model.
4. The `RequestBroker` also breaks up the query into subqueries to be made to different services. In a first step, it uses the constraints and the task concept property of the `MediatorComplexRequest` (in the example `GetMap`) to request a grounded, executable workflow from the Workflow Service (WS, see Specification 1.5).
5. When the Workflow assembly has been completed successfully, meaning that a Grounding Service has been found and additional `Transformer`s inserted as necessary for all preconditions, the workflow retrieved by the `RequestBroker` is passed on to the `TransformationQueueManager`, which manages execution of the workflow.
6. In parallel to step 5, the `RequestBroker` extracts all `AccessRequests` from the workflow, meaning requests to data provisioning services that need to be completed before a part of the processing can be executed. These `AccessRequests` are passed to the `AccessQueueManager`, which executes them and notifies the `RequestBroker` whenever one becomes available. The `RequestBroker` in turn notifies the `TransformationQueueManager`.
7. When all preconditions of a workflow managed by the `TransformationQueueManager` are satisfied, the process is started and run as far as all preconditions are available. When the process thus reaches it's finalization state, the `RequestBroker` is notified by the `Trans-`

formationQueueManager. The RequestBroker can then pick up the result and return it to the Interface Translation Façade which initiated the process, so that the ITF can encode the result according to the represented interface conventions and then send it back to the client.

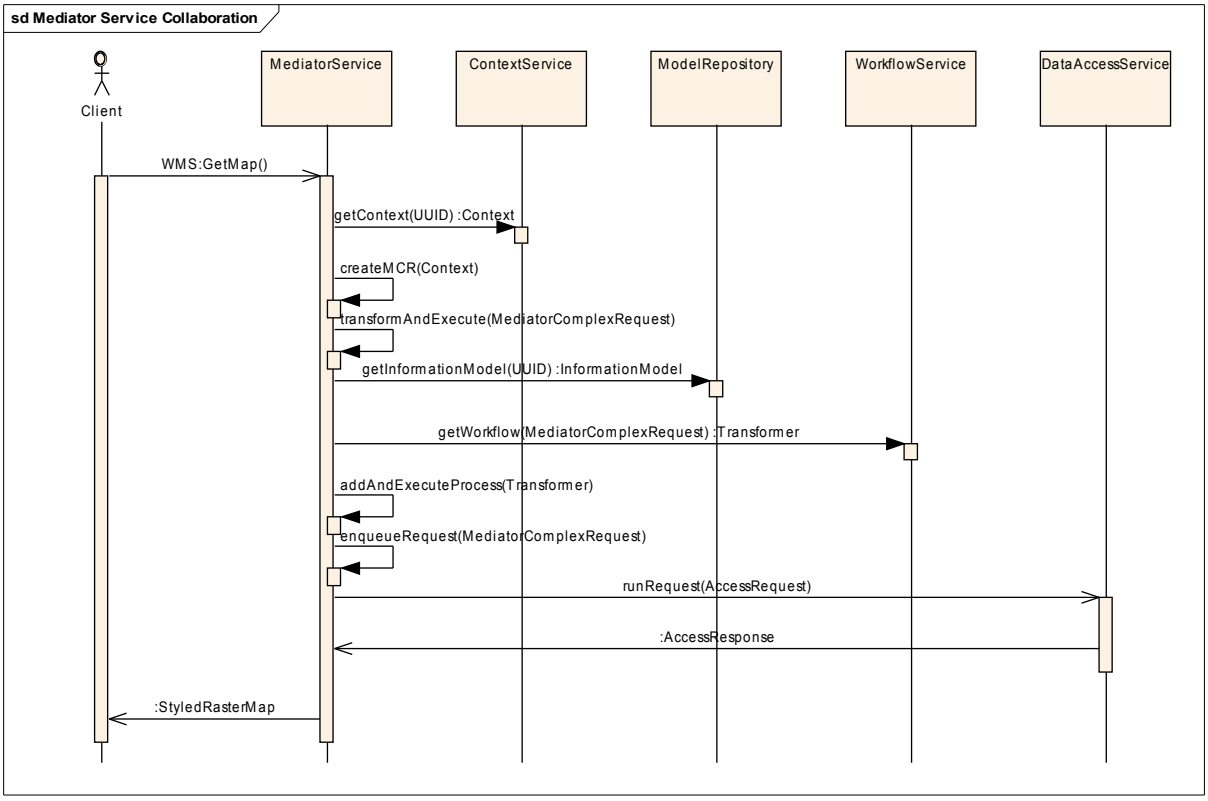


Figure 5: A Service-level sequence diagram outlining the interactions of the Mediator Service Component with other service components in the HUMBOLDT framework

3.2. Services offered by the Mediator Service

The Mediator Service indeed does not offer a new service in terms of the interface that it exposes to possible consumers; instead, it offers standardized service interfaces to trigger transformation and to access products created by transformation chains. All of these are well-defined through OGC standards. Currently, the Mediator supports the following interfaces:

- Web Map Service 1.1.1 and 1.3.0, with the following operations and encodings:
  - GetCapabilities, requested by URL KVP;
  - GetMap, requested by URL KVP, only raster formats (JPEG, PNG) returned.
- Web Feature Service 1.1.0, with the following operations and encodings:
  - GetCapabilities, requested by URL KVP;
  - DescribeFeatureType, requested by KVP;
  - GetFeature, requested by KVP, only mandatory parameters supported;
  - GetFeature, with Filter Encoding as a XML document via POST (required for parameter injection);
  - GetGmlObject, requested by KVP, only with mandatory elements;
  - All locking and updating of Features which the WFS interface allows are not supported.

- Web Processing Service 1.0.0, with the following operations and encodings:
  - `GetCapabilities`, requested by URL KVP;
  - `GetProcessDescription`, requested by URL KVP;
  - `Execute`, requested by posted XML document that directly contains a full product description.

For these services, only the mandatory operations are supported, and furthermore, only mandatory encodings are supported.

In addition to the parameters offered for the operations of these services, the Mediator Services may make use of Context UUIDs in string form in the host part of the URL used to invoke the service. Such a URL looks like the following example:

```
http://hostname.de:8080/mediator/550e8400-e29b-11d4-a716-446655440000/WMS?...
```

Of course, the hostname, port and path to the Mediator service might be adjusted in different deployments.

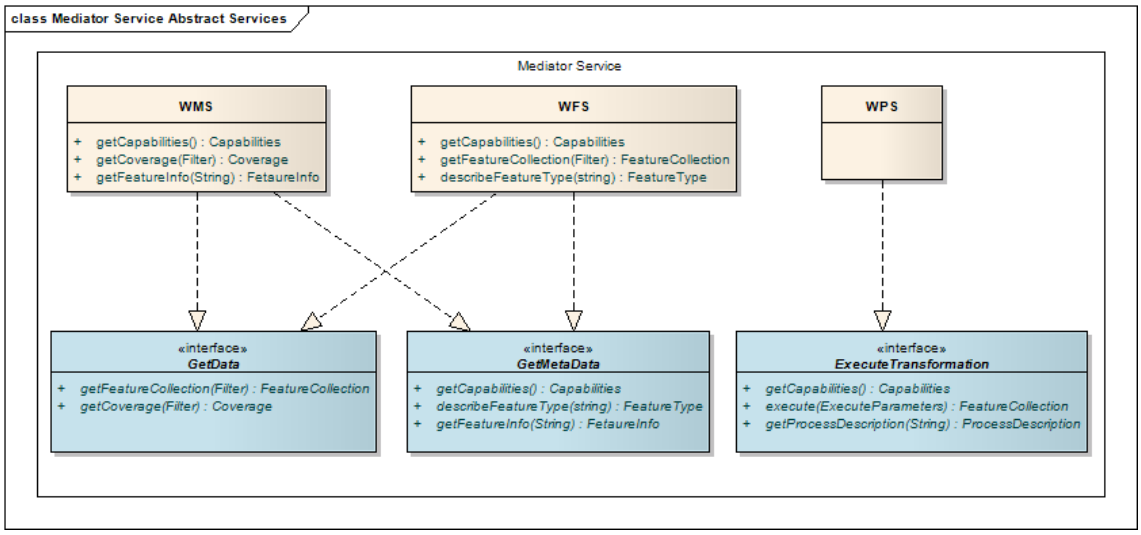


Figure 6: Abstract (blue, bottom) and concrete (rose, top) services that the Mediator Service offers.

Via these concrete service interfaces, three "*abstract services*" are provided, one providing access to data that has been transformed and harmonised (represented by the different `Get` operations, such as `WMS->GetMap`, `WMS->GetFeatureInfo` or `WFS->GetFeature`), the other one providing metadata via operations such as `WFS->DescribeFeatureType` and the `GetCapabilities` class of operations.

The final service, which is concretized in a WPS interface, represents the Mediator Service's capability as a complex transformation service or *"Invoke Spatial Data Services"* service. Here, all parameters that are required for a certain transformation are provided directly, without the need to work with other components such as the Context Service or the Workflow Service.

3.3. Internal Architecture of the Mediator Service Component

3.3.1 Interface Translation Façade

Full Name: Mediation Component Framework → Mediator Service Component → Interface Translation Façade (ITF)

1. Responsibilities of the Module

Provide the operations required by a certain service in the encoding(s) required by that service, such as `GetCapabilities` and `GetMap` for a WMS ITF implementation.

Decode the request into the internal request model, which consists of elements such as thematic constraints, spatiotemporal constraints and various metadata constraints.

- Enrich requests received via these operations with the context information that could not be transported via that interface, such as the identification of the used conceptual schema.
- Encode the response into the structure required by the implemented interface.

2. Collaboration

- The ITF accesses the Context Service (CS) when it has not received a complete context via the interface it implements.
- The ITF delegates the created complex request (`MediatorComplexRequest`) to the `RequestBroker`.
- The ITF makes use of the `ApplicationContext` module, which is provided by the container and manages the configuration of a certain instance of the Mediator.

3. Architecture overview

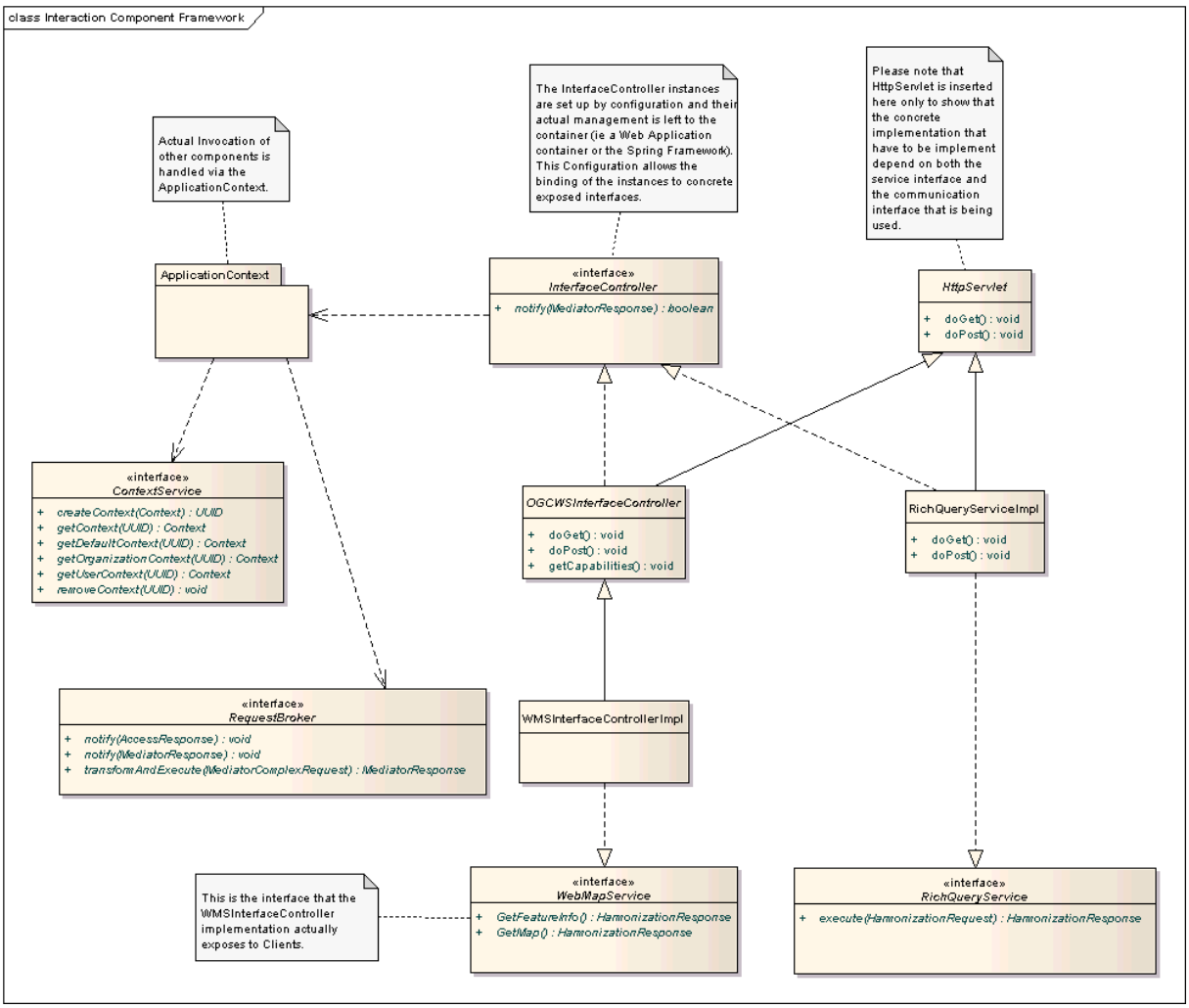


Figure 7: The main structure of the Interface Translation Façade, shown with example implementation classes for the case that the Façade is implemented as a `HttpServlet`.

Please note that the logical architecture of the ITF overall depends somewhat on the environment that it is implemented in. The ITF also has the purpose of isolating the rest of the Mediator Service form details of the implementation, such as the container, and from details and APIs specific to the used interfaces. This ensures that the requirement to be integrable into different environments (Service, Desktop Application) can be met.

4. API descriptions

The ITF does not have an API per se, but more a set of helper classes and marker interfaces. In the current version, the ITF implements the mandatory operations of WFS and WMS, as well as a few operations needed within the framework, as described below.

InterfaceController

`boolean`	`notify(MediatorResponse mres)` This operation is used when a `MediatorResponse` has been processed completely and can be encoded for sending to the client. It returns true if and when the `InterfaceController` implementation accepted and distributed the

	`MediatorResponse`.

3.3.2 Request Broker

Full Name: Mediation Component Framework → Mediator Service Component → Request Broker (RB)

1. Responsibilities of the Module

- The Request Broker is the main controller of a mediator service instance. It has knowledge of the operations that this instance can offer (via the Workflow Service), it verifies the request content (via the Information Model Service of the Model Repository), it can request access to data sources (services) and can apply transformations to data elements available to it, either as part of a request or a response.

2. Collaboration

- This module controls the `AccessQueueManager` to exchange data with data sources identified via the IGS.
- This module controls the `TransformationQueueManager` to apply necessary transformations to data retrieved from a grounding and to provide the requested answer satisfying the given constraints.
- This module calls the `WorkflowService` to retrieve a grounded workflow for the constraints specified in the `MediatorComplexRequest`.
- This module makes use of the Model Repository's `InformationModelService` for verification purposes.

3. Interface overview

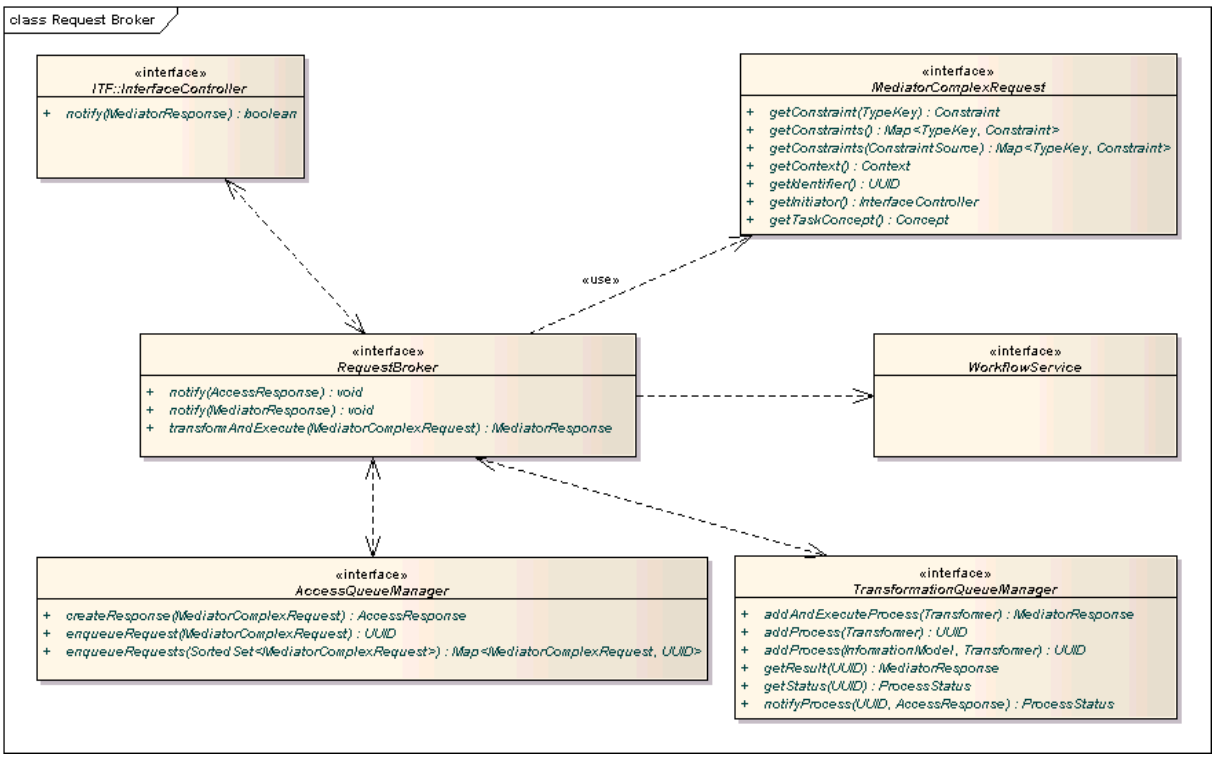


Figure 8: The main interface of the `RequestBroker` and of those components that it uses. Please refer to the API documentation for detailed description of the individual operations.

4. API Descriptions

The main operation descriptions for the `RequestBroker` are the following ones:

RequestBroker

`void`	`notify(AccessResponse ares)` This operation is used by the `AccessQueueManager` to tell the `RequestBroker` that a certain request has been completed and that the `AccessResponse` is complete.
`void`	`notify(MediatorResponse hres)` This operation is used by the `TransformationQueueManager` to notify the `RequestBroker` that a transformation/workflow was completed (i.e. reached its terminal state) and passes the result to the `TransformationQueueManager`.
`MediatorResponse`	`transformAndExecute(MediatorComplexRequest request)` This is the main method of the `RequestBroker`, which is being invoked by the ITF implementations to answer a client's request.

3.3.3 Access Queue Manager

Full Name: Mediation Component Framework → Mediator Service Component → Access Queue Manager (AQM)

1. Responsibilities of the Module

- The `AccessQueueManager` manages the asynchronous execution of all `AccessRequests` and notifies the `RequestBroker` when an `AccessResponse` is available.
- Via `DataAccessService` implementations, the AQM can also handle all physical schema translations necessary and also any protocol specialties required. A `DataAccessService` can be understood as an connector for upstream resources and can support very different types of data sources, ranging from files to services and relational databases.

2. Collaboration

- The AQM dispatches `AccessRequests` to the `DataAccessService(s)`.
- The AQM is being used by the `RequestBroker` to interact with data sources such as services.

3. Interface Overview

The following UML class diagram shows the AQM and the `DataAccessService`.

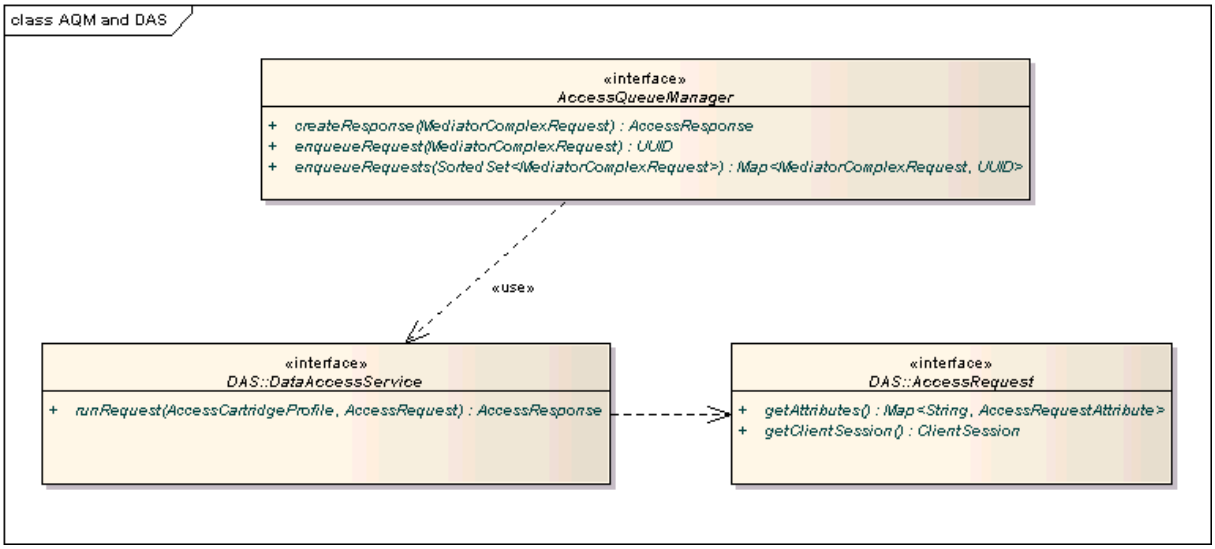


Figure 9: The interface of the Access Queue Manager and the Data Access Service

4. API Descriptions

AccessQueueManager

`AccessResponse`	`createResponse(MediatorComplexRequest)` Simpler API for cases where just one Request/response has to be managed.
`UUID`	`enqueueRequest(MediatorComplexRequest)` Enqueues one `MediatorComplexRequest` for asynchronous execution.
`Map<MediatorComplexRequest,UUID>`	`enqueueRequests(SortedSet<MediatorComplexRequest>)` Enqueues a `SortedSet` of `MediatorComplexRequests` for asynchronous execution.

3.3.4 Transformation Queue Manager

Full Name: Mediation Component Framework → Mediator Service Component → Transformation Queue Manager (TQM)

1. Responsibilities of the Module

- The `TransformationQueueManager` (TQM) is used by the `RequestBroker` to apply processing to various types of data at various stages. It is used both on the request path and on the response path, whenever a conceptual schema translation, geometric processing or other transformation is necessary. When working on response elements, individual `AccessResponses` are moved to the TQM by the RB. The transformations set up beforehand for which these `AccessResponses` represent the preconditions are then executed depending on the overall execution strategy and on pre- and postconditions of the individual transformations. Additional responsibilities of the TQM are:
- Discovery and Management of relationships between parts of a set of `MediatorComplexRequest`, i.e. application of a non-atomic Transformer only after all partial responses have arrived
- Delegation of `DataAccessRequests` to the `RequestBroker` (who will forward them to the `AccessQueueMananger`) if a transformer requires additional input that depended on other input.
- The component as a total is also responsible, together with the Transformers managed by it, to ensure the creation of a log of the applied processing steps, the so-called `Lineage`. For each dataset transformed in the TQM, such a lineage consisting of Processing steps, each described by identifier and with all parameters that were put into it, must be created. This not only allows a detailed evaluation of the processing by expert users but also allows repeatable execution and partial re-execution of a workflow.

2. Collaboration

- The TQM accepts workflows to be processed that are dispatched by the `RequestBroker`.
- The TQM notifies the RB when a `Transformation/Workflow` is completed.
- The TQM invokes individual Transformers, which can be local instances or representative of remote service instances.
- Individual Transformers may make use of the `ModelMappingService` of the Model Repository.

3. Interface Overview

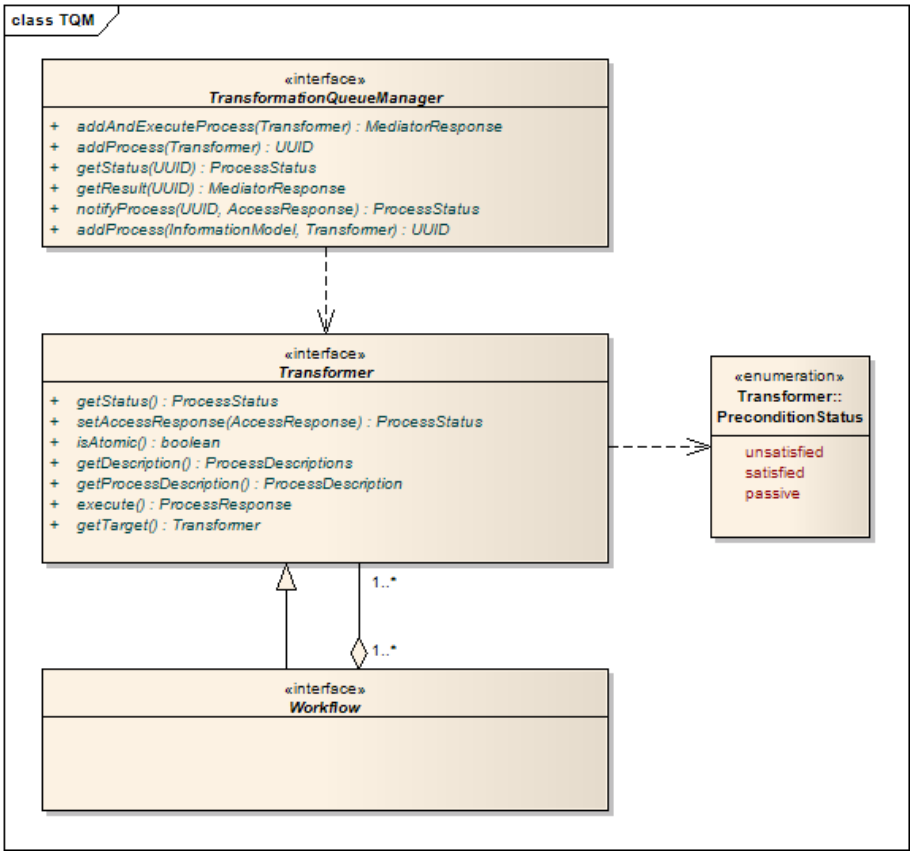


Figure 10: The interface of the Transformation Queue Manager and some of it's main dependencies

4. API Descriptions

TransformationQueueManager

`MediatorResponse`	`addAndExecuteProcess(Transformer)` This operation invokes a synchronous execution of the given transformer.
`UUID`	`addProcess(InformationModel, Transformer)` This method is a convenience method to add a simple, one-step transformation of an `InformationModel` to the TQM.
`UUID`	`addProcess(Transformer)` This method adds a Transformer/Workflow for asynchronous execution.
`MediatorResponse`	`getResult(UUID)` Returns the `MediatorResponse` containing the result of the transformation process specified by the given UUID.
`TQM.ProcessStatus`	`getStatus(UUID)` This operation is used to request the processing status of a certain process.

`TQM.ProcessStatus`	`notifyProcess(UUID, AccessResponse)` Notifies the TQM that for the process identified with the given UUID, an `AccessResponse` has been retrieved and is available. Returns the `ProcessStatus` that the identified process is in after assigning the precondition.

4 Information Viewpoint

This chapter describes the information viewpoint of the Mediator Service Component as defined by the RM-ODP. This covers the various data structures that are being used for storage and exchange between the service components defined in the computational viewpoint.

For the Mediator Service, the main data structure is the Mediator Complex Request (MCR), which is described in the following sections. Please note that many of the Constraints are also explained in the Context Service Component specification and are therefore referenced only. The second main structure is the Mediator Container Model (MCM), a data structure into which the geodata and non-spatial data read is abstracted for processing. This MCM is to large extents built from existing standards, such as GML 3.2.1. Finally, the third main data structure is the HUMBOLDT metadata profile, an adoption of ISO 19115 that formalizes some aspects like identification and lineage in a more stringent way.

4.1. Mediator Complex Request (MCR)

The MCR is described in detail in the HUMBOLDT Commons specification, since it 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 and WCS, and is enriched by information from a Context document, if available.

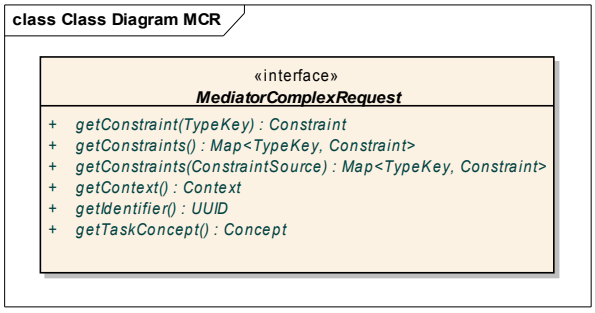


Figure 11: The core MCR interface.

As can be seen from Figure 11, the MCR has four key elements that define it:

- *A Set of Constraints:* Each Constraint 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 detailed in the Context Service Component Specification and also sown in Fehler: Referenz nicht gefunden Fehler: Referenz nicht gefunden. Some examples are:
  - *ThematicConstraint*: Defines the requested Feature Types or more conventionally Layers by giving their identifiers from a controlled vocabulary.
  - *SpatialConstraint*: Defines the spatial area from which geoinformation is needed, either as an `Envelope` or as a more complex `Geometry`.
  - *TemporalConstraint*: Defines the temporal extent of the spatial information to be retrieved.
  - *PortrayalConstraint*: Defines portrayal rules for the data set to be retrieved.

  - *ServiceConstraint:* Defines a type of service to use upstream. This constraint can be used to limit the automatic selection of services to use down to a single concrete service or to a certain type of services.

- *A Context*: The context document valid for this request. The context is the second main source for Constraints apart from the original request itself. It can either be a user-specific Context, and organization-specific Context, or a default Context if no information was included in the original request. All Constraints from the Context are merged with those from the original request, using precedence rules as defined by the system administrator. This can involve setting blocking Constraints (ones that may not be overridden, typically done on organization level). If a Constraint from the Context is non-blocking, it will be overwritten by a Constraint of the same `ConstraintType` from the original request.

- *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 Task Concept*: This element is used to identify the operation a user requested. A task concept is in the first line a representation of the goal that can be achieved by calling a certain operation on one of the interfaces offered by the Mediator Service – it is thus dependent on the operation and the interface that was invoked, such as `GetMap` on a WMS. In addition, the task concept can be refined further by making the selection of a task concept can also depend on request parameters. As an example, for the WMS `GetMap` operation with a file type of `img/jpeg`, the matching task concept would be `GetStyledRasterMap`. All Task Concepts together form the System Task Taxonomy (STT)[4]. The approach of using a System Task Taxonomy allows the definition of basic workflows that can be extended and modified for more specific tasks, adopting strengths of the object-oriented programming paradigm for workflow management and execution. For more information on the STT and the Task Concept, please refer to the Workflow Service Component Specification.

4.2. Mediator Container Model (MCM)

To be able to use a single set of harmonisation processors on a set of geographic data instances, first a unified model for the data to be processed needs to be established. This unified model consists of several areas, each of which is described in the following sections and can be understood as a Level 1 meta-model according to the definitions of the OMG MOF (Object Management Group Meta-Object Facility).

The individual sections have a common meta-model, which is also used in WP07 (data harmonisation) and which is based on the OMG MOF (Level 2). This makes it easier to use established transformation languages and to use existing tools, both for data modelling and for transformations.

4.2.1 Conceptual Schema

For the expression of conceptual schemas, UML and specifically GML Application Schemas built on UML, as described in ISO 19101 and others are directly used. For detailed information, please refer to the specification of the HUMBOLDT Model Editor (Component [2.3.1]). The Figure below shows the

4 In scientific literature, the term System Task Ontology (see Fabrikant, s. (2001), *Building Task-Ontologies for GeoVisualization*, in: Proceedings of ICA Commission on Visualization and Virtual Environments Pre-Conference Workshop on Geovisualization on the Web, Beijing, China, 2001) is usually used. However, since we wanted to mostly leverage is-a relations and wanted to simplify processing rules, we have decided to narrow down the strucutre to a taxonomy of tasks.

metamodel for this conceptual Schema Language. In addition to this metamodel, the CSL contains a typing system for the precise definition of attributes and especially of geometric attributes.

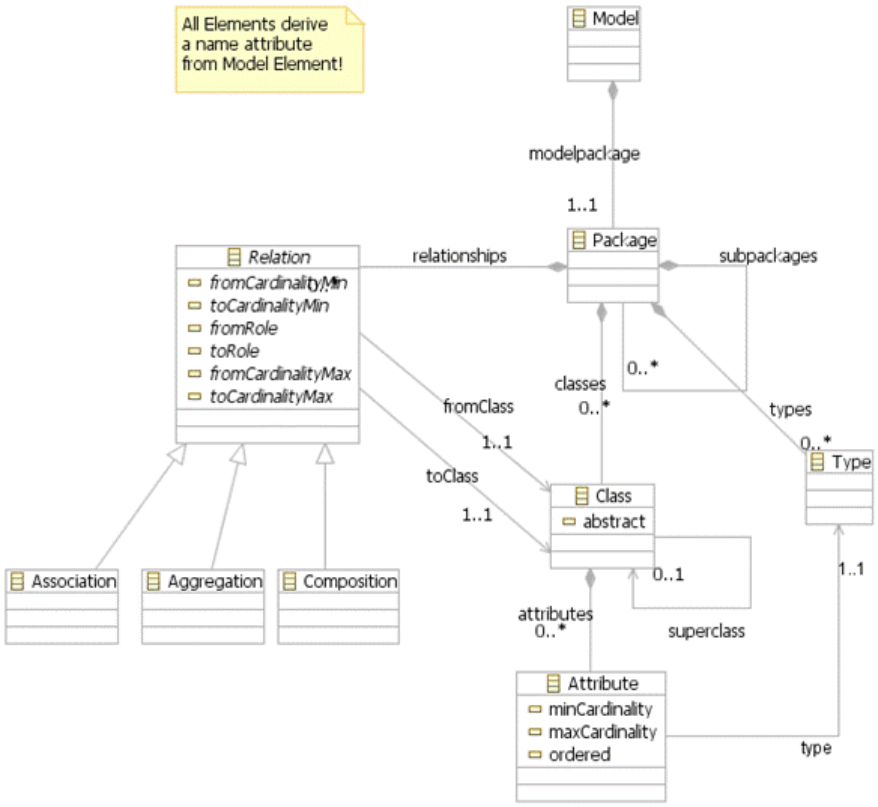


Figure 12: Metamodel of the main elements of the Conceptual Schema Language used

It is being evaluated whether specific Ontology elements should be added to this model.

For the Mapping of Conceptual Schemas, the OMWG's Ontology Mapping Language (D7.2) is used. Please refer to the [2.3] specification for details.

4.2.2 Feature Model

To describe Features and Feature Collections, both of vector geometry and of coverages, GML's (ISO 19136) Feature model is used.

These are the Conformance classes of the Feature Model in the MCM, as described in the GML 3.2.1 specification:

GML application schemas defining features and feature collections *A.1.4*

It should be noted that all `FeatureTypes` used are `FC_FeatureTypes` in the sense of ISO 19110 and as described in section FIXME of the Model Repository Service Component Specification..

4.2.3 Geometry Model

As the main model for 0D to 3D vector geometry, GML's (ISO 19136) Geometry model is used, which in turn is based on ISO 19107.

These are the Conformance classes of the Geometry Model in the MCM, as described in the GML 3.2.1 specification:

GML application schemas defining spatial geometries *A.1.5*

It should be noted that for M2.0, only those sub-clauses of the abstract test suite dealing with 0D/1D/2D data are to be fulfilled.

Consequently, an open point for later versions is the inclusion of specific 2D/3D Geometry types that are used in the CAD/DCC areas, such as NURBS. A candidate standard for alignment in this area is COLLADA.

4.2.4 Coverage Model

These are the Conformance classes of the Coverage Model in the MCM, as described in the GML 3.2.1 specification:

GML application schemas defining coverages *A.1.9*

It should be noted that for M2.0, only Grid coverages (sub-clause of the abstract test suite A.2.15.6) are to be implemented, as coverages have not been used extensively so far. A finer specification of these is one of the goals for the next version of the specification (M3.0), based on the results of in-depth work with the Ocean scenario.

4.3. HUMBOLDT metadata profile (HMD)

Metadata elements, as defined in ISO 19115, play a central role in many parts of the HUMBOLDT framework. They are required to identify whether a certain data set can fulfil the constraints imposed by a user or not, and they are required to provide traceability of harmonization and transformation processes. Compared to the Core profile of ISO 19115, the following additions are made:

- Additions to cover requirements stated in the INSPIRE Generic Conceptual Model (Document 2.5) and from INSPIRE's Draft Implementing Rules for Metadata, specifically Identification by namespace, version and unique identifier (Requirements 70 to 72)
- Machine-readable specification of Lineage and Data Quality elements, as being critical to cost/benefit evaluation during automated service composition and for other purposes

A HUMBOLDT thematic working group is working on the topic to provide a model with complete requirements coverage for M3.0 of the specification.

4.3.1 Identification and Citation

In a spatial data infrastructure, especially when it comes to providing transaction capabilities, an unanimous identification is essential. For this reason, the Identification element of ISO 19115 is profiled with the following mandatory elements:

HMD_Identification

`InternationalString`	`getTitle()` Returns the Name by which the cited resource is known. → INSPIRE Document 2.6, Section 2.2.1.1
`InternationalString`	`getAbstract()` Returns a brief narrative summary of the content of the resource(s), as a free-text string. → INSPIRE Document 2.6, Section 2.2.1.2

`URI`	`getNameSpace()` Returns the namespace of the resource, as an URI. → INSPIRE Document 2.6, Section 2.2.1.4
`UUID`	`getIdentifier()` Returns a UUID that will usually suffice for uniquely identifying a resource. Compared to INSPIRE, where free text is allowed, we are using UUIDs. Internally, an optional local identifier, which is free-text, can be used in addition. → INSPIRE Document 2.6, Section 2.2.1.5
`int`	`getVersion()` Returns the version of the resource, as a single unsigned `int` value.
`ResponsibleParty`	`getResponsibleParty()` Returns the `ResponsibleParty` of the resource, as defined in ISO 19115.

`Identification` should be used on the level of Features, but not below, since it is rather heavy-weight and not really suitable for usage with finely grained objects such as large point clouds.

It should be noted that the optional elements specified in ISO 19115 can be added whenever required, and that the above elements have to be considered as mandatory. Compared to the INSPIRE Metadata IR, the Identification also omits the resource type, as this aspect's value domain is quite underspecified currently and will be included in a more detailed description later on.

Furthermore, it can be seen that the ISO elements of Citation and Identification have more or less been combined, as it has also been done in INSPIRE. If a split is to be done in later versions, the Citation should focus on the responsible party, whereas the Identification should uniquely identify the resource.

4.3.2 Data Quality and Lineage

In the HUMBOLDT Framework, each Transformer adds a step to the processing history of a data set, possibly improving the quality of the data set, but possible degrading it for a different purpose. Therefore, it is important to always have the information ready about what has happened to a data set before. To this purpose, the Lineage elements of ISO 19115 are formalized so that they can be generated and read by the Framework. This section explains how that is done in a first draft.

HMD_Lineage

`Collection <HMD_ProcessStep>`	`getProcessSteps()` Returns a collection of descriptions about an event in the creation process for the data set in question. → ISO 19115:2003, Section B.2.4.2.1, 84
`Collection <Source>`	`getSources()` Returns information about the source data used in creating the data specified by the scope. The `Source` element is left as it is described in ISO 19115. → ISO 19115:2003, Section B.2.4.2.1, 85

`InternationalString`	`getStatement()` General explanation of the data producer's knowledge about the lineage of a dataset. → ISO 19115:2003, Section B.2.4.2.1, 83

The `ProcessStep` element also has the same structure as in ISO 19115, but the value domains change, as explained below.

HMD_ProcessStep

`String`	`getDescription()` Returns a collection of descriptions about an event in the creation process for the data set in question. The returned `String` is not free text, as in ISO 19115, but structured as an XML document with the XML Schema found in Annex A.2. You can also find an UML diagram of the Schema there. → ISO 19115:2003, Section B.2.4.2.2, 87
`URL`	`getProcessor()` Returns the identification of the processor used, expressed as the URL pointing to the Web Processing Service operation used. If the processor used is an internal one, provide it's class name with a local:// prefix. → ISO 19115:2003, Section B.2.4.2.2, 90
`Collection <Source>`	`getSources()` Returns a collection of information on the source data sets used for the processing in this step. → ISO 19115:2003, Section B.2.4.2.2, 91

5 Engineering viewpoint

The engineering viewpoint focuses on mechanisms for distribution, distribution transparencies and support services such as security and persistence.

5.1. Service to physical architecture mapping

The Mediator Service is designed and implemented as a service without shared state across multiple instances. It is thus usually deployed to a single virtual machine on a single physical machine, but can also be deployed to a VM spread across multiple physical machines or be distributed on multiple VMs.

All modules of the Mediator Service normally form just one service. However, some of the Transformers can be remote processing services that are self-sustaining units. This is also shown in the following Figure.

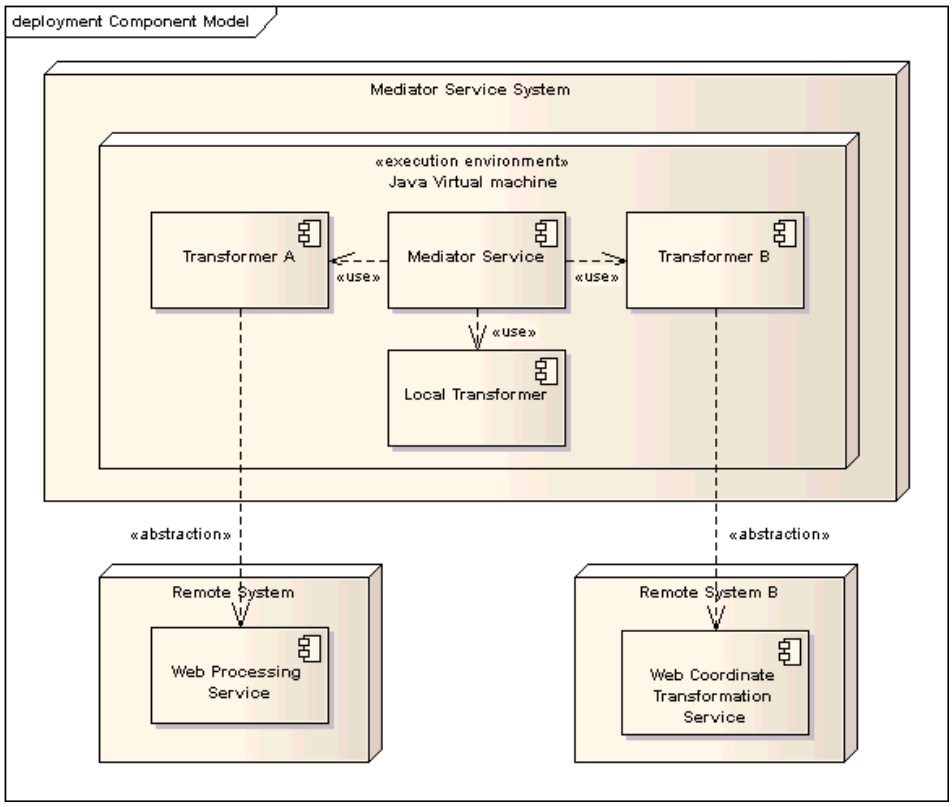
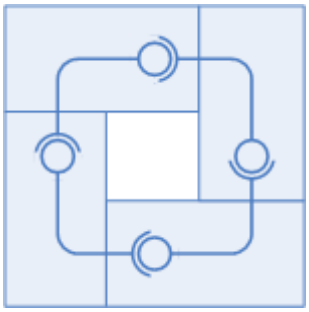


Figure 13: The Logical to Service and physical system mapping, showing that some processing services are services of their own both physically and logically

The reason for this mapping of logical components to composites were mainly the following:

1. Efficient use of runtime resources: Where it is expected that a lot of data has to be exchanged, tighter bundling was aimed for, with the following exception:
2. Some of the components are going to be implemented based on existing standalone software, specifically the Context Service, the Grounding Servers including the Catalogue Web Service and the Information Grounding Service.

6 Summary & Outlook



With initial feedback from the implementation and usaage of ther service in a scenario, the specification that is presented with this document can be understood as a solid basis for one of the core services of the HUMBOLDT framework.

Of course, there are quite a few aspects which can be improved and will have to be improved in the next two versions of this specification. This is the list currently known:

M3.0:

- Include more precise descriptions of the handling of live streamed data ans sensors integration.
- Incorporate final results of HUMBOLDT Metadata Thematic WG
- Include more precise descriptions for Coverages and Web Coverage Service Support

Annex A: Schemas

Workflow Exchange Schema

This schema is used by the Mediator Service and the Workflow Service to exchange simple workflows.