Testbed-12 Data Broker Engineering Report

Table of Contents

Publication Date: 2017-06-30

Approval Date: 2016-09-13

Posted Date: 2016-10-25

Reference number of this document: OGC 16-045r2

Reference URL for this document: http://www.opengis.net/doc/PER/t12-E002

Category: Public Engineering Report

Editor: Daniel Balog, Robin Houtmeyers

Title: Testbed-12 Data Broker Engineering Report

Testbed-12 Data Broker Engineering Report (16-045)

COPYRIGHT

WARNING

This document is an OGC Public Engineering Report created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard and not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

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

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

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

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

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

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

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

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

Abstract

An important principle of a Service Oriented Architecture (SOA) is the notion of composing capabilities provided by individual services into complex behavior. A requester should be able to compose a solution using functionality or data offered by multiple services without worrying about underlying differences in those services.

Each OGC service is designed to offer a specific type of data product via a service-specific interface. This Engineering Report (ER) describes a single service interface that allows access to multiple data sources, possibly heterogeneous with respect to the types of data provided.

This report advances the work started in OGC Testbed 11 with the addition of heterogeneous data sources, as well as several other enhancements.

Business Value

Modern-day geospatial architectures consist of a multitude of geospatial services, each dedicated to serve a specific data type to its service consumers. This often comes with separate services to cover different areas of interest, leading to additional complexity for the service client.

The Data Broker addresses this complexity by acting a single endpoint, in a hierarchy of OGC services with heterogeneous data.

The business value is that using a Data Broker can drastically simplify the workflow required to look for, request and visualize data from the perspective of a client. This has a benefit for both architecture designers looking for a way to offer a set of services as a single endpoint, as well as client developers who will be able to write simpler applications with less coupling between their applications and the various OGC services involved in the architecture.

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

OGC services make it possible to set up an interoperable, standards-based geospatial architecture covering a variety of data types and capabilities. In practice, complexities can arise for the user to interact with the resulting architecture: in case of multiple services covering different areas of interest, in case of heterogeneous data offerings, in case of services with different capabilities, etc.

The Data Broker offers a way to relieve some of the complexity by specifying

and describing an architecture that allows hierarchies of OGC services, such as WMS, WFS and WCS, to be offered as a single endpoint.

The Data Broker does this to simplify interactions with the client and making it more obvious for systems integrators to design an architecture using OGC services. A core aspect of this is provenance metadata, conflation, caching and the use of catalogue services to dynamically detect data.

How does this ER relate to the work of the Working Group

The Workflow DWG is always looking for ways to improve workflow orchestration. This ER defines a few use cases specific to the domain of Aviation, where workflow orchestration can simplify interactions between client and service. The lessons learned while developing the Data Broker, and this resulting ER can prove to be a valuable starting point for generalization of other workflows using OGC Services.

Besides simplification, another goal of Workflow DWG is to provide adequate metadata about subjects such as lineage. This ER discusses how lineage metadata can be added to various OGC Services. While some of the solutions are specific to the Aviation domain, the conclusions also apply for other domains.

Keywords

ogcdocs, testbed-12, brokering, data broker, mediation, semantics, aggregation, WFS, WMS

Proposed OGC Working Group for Review and Approval

Workflow DWG

Chapter 1. Introduction

1.1. Scope

The purpose of this report is to research various topics related to Data Brokering in the Aviation thread. The report shall:

- Describe the use of a Data Broker providing multiple web services (WFS, WMS).
- Describe how to handle mediation between data formats with semantically equivalent data.
- Describe the interaction between a Data Broker and Catalogue Service for endpoint and feature discovery.
- Document potential enhancements and identified issues for relevant documents and references.

1.2. Document contributor contact points

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

Table 1. Contacts

Name	Organization
Daniel Balog	Luciad
Robin Houtmeyers	Luciad

1.3. Future Work

Improvements in future testbeds are desirable to the following topics:

- Integration of an asynchronous messaging capability to support publish-subscribe messaging of aviation data through the Data Broker. Research on the core topic of asynchronous messaging for aviation data has been conducted during Testbed 12. The result of this research is described in a dedicated ER [6].

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- [OGC 06-121r9] OGC® Web Services Common Standard
- [OGC 13-131] OGC Publish/Subscribe Interface Standard 1.0 - Core
- [OGC 13-133] OGC Publish/Subscribe Interface Standard 1.0 - SOAP Protocol Binding Extension
- [OGC 15-028] OGC Testbed 11 Data Broker Specifications Engineering Report
- [OGC 16-017] OGC Testbed 12 Asynchronous Messaging for Aviation
- [OGC 16-028] OGC Testbed 12 FIXM GML Engineering Report
- [OGC 09-025r2] OGC® Web Feature Service 2.0 Interface Standard
- [OGC 06-042] OpenGIS Web Map Service (WMS) Implementation Specification
- [OGC 12-168r6] OGC® Catalogue Services 3.0 - General Model

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9] and in OGC® Abstract Specification [OGC 08-126] shall apply. In addition, the following terms and definitions apply.

3.1. Abbreviated terms

AIXM - Aeronautical Information Exchange Model
AMDB - Aerodrome Mapping Data Base
AMXM - Aerodrome Mapping Exchange Model
AMXS - Aerodrome Mapping Exchange Schema
API - Application Programming Interface
BBOX - Bounding Box
CCI - Cross-Community Interoperability
COTS - Commercial Off the Shelf
CRUD - create, read, update, delete
CSW - Catalog Service for the Web
DNOTAM - Digital NOTAM
ECQL - Extended Common Query Language
EFB - Electronic Flight Bag
FIXM - Flight Information Exchange Model
FNS - Federal NOTAM Service
FPS - Feature Portrayal Service
GML - Geography Markup Language
HTML - HyperText Markup Language
HTTP - HyperText Transfer Protocol
ICAO - International Civil Aviation Organization
ISO - International Organization for Standardization
JDK - Java Development Kit
NOTAM - Notice to Airmen
OGC - Open Geospatial Consortium
RDF - Resource Description Framework
SBVR - Semantics of Business Vocabulary and Business Rules
SOA - Service Oriented Architecture
SQL - Structured Query Language
SWIM - System Wide Information Management
UUID - Universally Unique Identifier
WCS - Web Coverage Service
WFS - Web Feature Service
WFS-T - WFS-Transactional
WMTS - Web Map Tile Service
WPS - Web Processing Service
WXXM - Weather Information Exchange Model
XMI - XML Metadata Interchange
XML - Extensible Markup Language
XSLT - Extensible Stylesheet Language Transformations

Chapter 4. Overview

This Engineering Report (ER) builds on top of the Data Broker defined in OGC Testbed 11, which primarily focused on brokering of AIXM 5.1 data through OGC WFS [1]. Chapter 5 and 6 define the status quo of the Data Broker at the start of Testbed 12, the new requirements identified within Testbed 12 and the proposed solutions to address them. Before the ER delves deeper into the technical aspects of the requirements and solutions, Chapter 7 provides an overview of the Broker's architecture and targeted uses cases.

A primary new Data Broker capability looked at within Testbed 12 is portrayal. Chapter 8 investigates how the Data Broker can be extended with portrayal capabilities to improve the interoperability with clients not supporting OGC WFS and / or aeronautical data exchange formats. Additionally, Chapter 10 investigates how the Data Broker's WFS response can be extended with portrayal information.

Next to AIXM 5.1, the aeronautical exchange format AMXM 2.0 is added to the Data Broker's list of input and output formats supported. Because both formats are semantically equivalent, the Data Broker could use mediation to combine data from heterogeneous but semantically equivalent data sources. The implications of this on the Data Broker's architecture are discussed in Chapter 9, along with an overview of an evaluation of AMXM 2.0 in the light of the Data Brokering use case.

Finally, Chapter 11 discusses the topic of dynamic data discovery, by means of integrating with a Catalogue Service. This enables a Data Broker to automatically find relevant data sources and web services, instead of requiring manual configuration.

Chapter 5. Status Quo & New Requirements Statement

5.1. Status Quo

In OGC Testbed 11, a significant effort was made to research and develop the core specification of a “Data Broker” concept for the Aviation thread, based on OGC technology. The overall goal was to enable the setup of cascading OGC web services to form a data source chain, in which one service is capable of providing information coming from other services.

The Data Broker implemented in OGC Testbed 11 was a pure WFS 2.0 that aggregated its data on-the-fly from other data sources. Clients connecting to this service could not distinguish it from a regular WFS 2.0 service.

The following topics were analyzed and implemented:

- A Data Broker that chains aggregates data from multiple WFS sources, while performing conflation (to avoid data duplication).
- A study was carried out on the performance of chaining together a hierarchy of WFS 2.0 services, using several caching techniques.
- Data enrichment for aggregated data originating from a different source, containing provenance information of the data.

5.2. Requirements Statement

Building on top of the previous testbed’s report and implementation, the following new functional requirements have been stated and will be analyzed in this testbed:

- Offering heterogeneous services in the Data Broker, such as WFS and WMS, in a singular, uniform service that offers a single point of entry.
- Mediation of data that is semantically the same, but is served in a different format. More specifically, an analysis will be performed for the AMXM 2.0 and AIXM 5.1 formats.
- Research on embedding portrayal information in WFS responses from the Data Broker.
- Use of the CSW Catalogue Service for endpoint and feature discovery.

Chapter 6. Solutions

6.1. Targeted Solutions

Building on OGC Testbed 11, various targeted solutions were investigated and discussed during Testbed 12. At a high-level, these targeted solutions were:

- Implement WMS support in the Data Broker to act as a portrayal service
- Add feature metadata to WMS requests that are the portrayed result of feature data. This allows clients to get more information about what is being portrayed.
- Use the CSW Catalogue Service for endpoint and capabilities discovery
- Add portrayal information to WFS requests. This allows clients to perform portrayal using the Data Broker without having to worry about styling.
- Semantically mediate AMXM 2.0 to AIXM 5.1 for clients that are not capable of parsing AMXM 2.0 data

Although not part of the requirements, the Data Broker developed in Testbed 12 also included brokering of FIXM 4.0 (with GML) data.

6.2. Recommendations

Here is a summary of the recommendations identified during Testbed 12. For details, refer to the following chapters.

- Extend WMS to support OGC Filtering to enhance portrayal capabilities.
- Add an ISO 19115 metadata property to AMXM 2.0 features, to be able to add lineage and other metadata information when brokering individual features.
- Add an ISO 19115 metadata property to a WFS FeatureCollection type, to allow describing portrayal and other metadata information (such as portrayal endpoints) to any type of data requested from a WFS service.

Apart from these recommendations, a change request has also been identified for FIXM 4.0, i.e. the integration of an optional ISO 19115 / 19139 metadata property on a Flight feature. The need for this was revealed during the Data Broker's data aggregation process, to allow the integration of lineage / provenance information on FIXM flight features. For details, refer to the FIXM GML Engineering Report, OGC 16-028.

Chapter 7. Architecture and Use Cases

This chapter gives an overview of the Data Broker use cases defined in Testbed 12, along with an overview of its architecture and implementation. Subsequent chapters delve deeper into these use cases, with a discussion on how to address them in the Data Broker's architecture.

7.1. Architecture

The high-level architecture of the Data Broker in Testbed 12 is shown in the figure below. Similar to Testbed 11, it connects to WFS providers offering AIXM 5.1 data. New is the addition of providers offering AMXM 2.0 and FIXM 4.0 data, an integrated WMS and integration with a CSW to discover data providers.

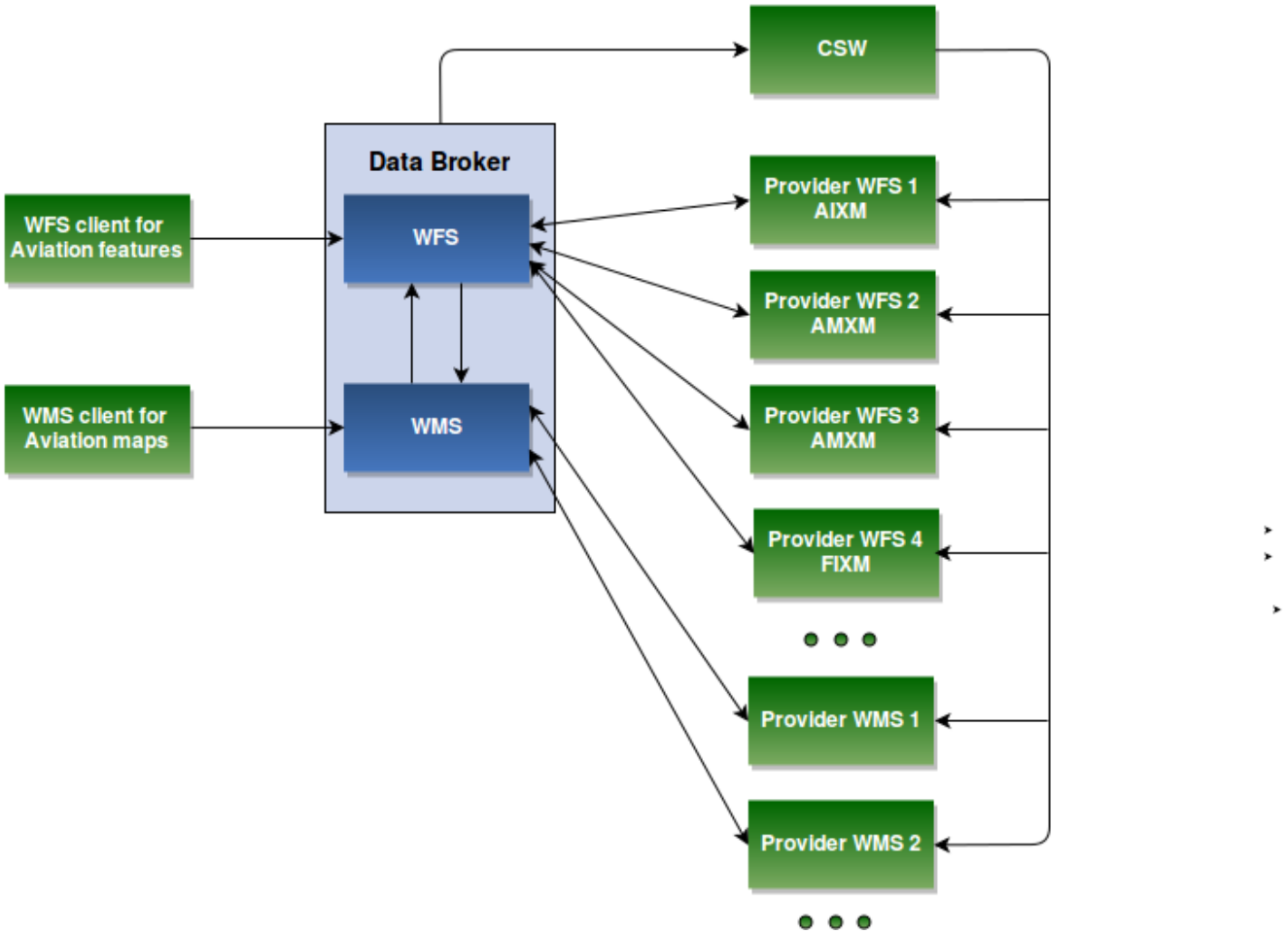

Figure 1. Architecture of the Data Broker

On an interface level, the data broker is split up into two services: WMS and WFS. Both services offer different access to the same data, and the individual services can communicate between each other to exchange data where needed. It is possible for instance to request a layer from the WMS service that is a portrayed version of the feature types of a WFS service. This allows both WFS and WMS based clients access to the same data.

On the business level, the data broker uses a CSW for endpoint discovery. The CSW provides the Data Broker with a list of AIXM, AMXM and FIXM based WFS services, as well as existing WMS services. The list could even include other Data Brokers. The Data Broker uses the capabilities document of each of the services to aggregate and conflate the capabilities of all the services it

mediates. This means that for each of the feature types available in each of the source WFS services, a feature type is created in the capabilities document of the Data Broker. If multiple WFS services are found with the same feature types, then their capabilities documents are merged, taking into account the enlarged bounds of the feature types.

For WMS, a portrayal layer is created for each of the WFS feature types available in the Data Broker. This allows clients without WFS capabilities to display feature types without any GML decoding abilities. Finally, the WMS also contains a layer for each of the layers inside an aggregated source WMS service.

7.2. Implementation

To support the Data Broker research in Testbed 12, the implemented Data Broker of Testbed 11 was extended based on the use cases and architecture discussed in the previous sections. This particular implementation was built on top of Luciad's COTS software product LuciadFusion. LuciadFusion offers a set of standards-based software components for server-side development, including an OGC Web Services Suite equipped with OGC-compliant WFS and WMS service components. One of these component's benefits for the Data Broker task is its open data back-end API, allowing users to easily connect to any type of storage component – such as other OGC web services.

7.2.1. Functional overview

The implemented Data Broker has the following functionality:

- OGC-compliant WFS 1.1.0 & 2.0.0 service interface with support for the following requests: GetCapabilities, DescribeFeatureType and GetFeature. Supported request encodings are HTTP GET and POST.
- OGC-compliant WMS 1.1.1 & 1.3.0 service interface with support for the following requests: GetCapabilities, GetMap, GetFeatureInfo. Support is also provided for the WMS' Styled Layer Descriptor profile: users can supply an SLD with user-defined layers and styles. Supported request encodings are HTTP GET and POST.
- Data broker capabilities
  - Support for brokering of OGC WFS data sources complying with versions 1.1.0 and / or 2.0.0 and supporting data exchange formats AIXM 5.1, AMXM 2.0 and / or FIXM 4.0 (with GML).
  - Automated data source discovery by means of OGC CSW integration.
  - Conflation based on unique identifiers.
  - Aggregation of similar features types from different OGC WFS data sources in to one feature type.
  - Semantic mediation between AIXM 5.1 and AMXM 2.0 feature data.
  - Addition of provenance by integrating lineage information on the feature type level in the capabilities and on the feature level by adding ISO 19115-based lineage metadata to AIXM 5 features.

7.3. Deployment characteristics

The Luciad Data Broker implementation is based on Java Servlet technology. To run, the Broker requires a Java servlet container or application server compatible with Java Servlet 3.0 or higher. Apache Tomcat 7 was used during Testbed 12. Other than being capable of running a Java Virtual Machine 1.7 (or higher) and an appropriate servlet container / application server, no requirements are posed on the underlying hardware or operating system.

7.4. Use cases

7.4.1. Use case 1: Requesting maps from the Data Broker

The first use case relates to the ability to request maps from the Data Broker, next to feature data. The figure below shows a sequence diagram for this use case, taking the example of requesting a map backed by AIXM 5.1 data.

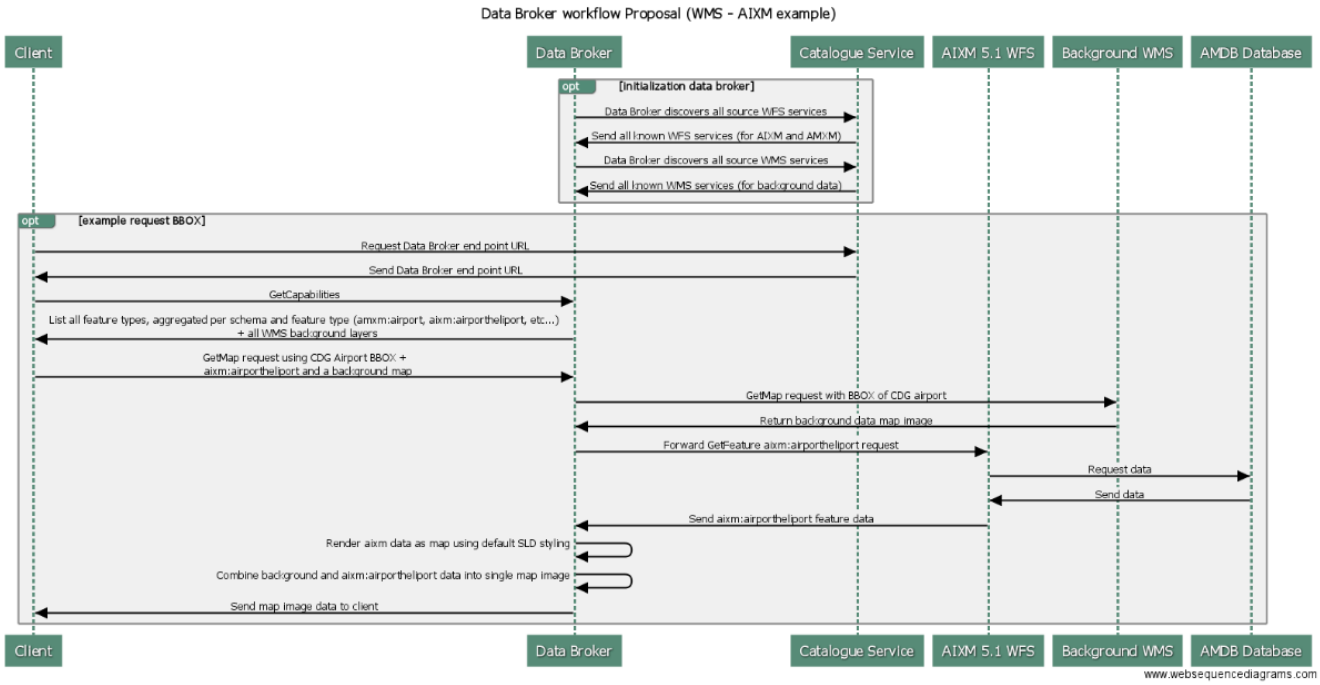

Figure 2. Request AIXM 5.1 data as Map, using WMS

On a high level, the following steps take place:

1. As initialization for the Data Broker, it will connect to the CSW to retrieve all possible WFS and WMS endpoints
2. As initialization for the client, it will connect to the CSW to retrieve the Data Broker endpoint
3. The client gets the WMS capabilities of the data broker. In this process it will get an overview of all the layers that are aggregated from AIXM, AMXM and FIXM feature-types, as well as WMS sources that offer background datasets.
4. The client performs a GetMap request on a WMS layer that aggregates AIXM 5.1 airport locations, along with some background satellite imagery.
5. The data broker retrieves the satellite imagery from a WMS service

6. The data broker then retrieves the AIXM 5.1 feature data from one or more of its aggregated WFS services.
7. The data broker uses the AIXM 5.1 feature data for portrayal purposes, using default SLD styling.
8. The satellite imagery and portrayed AIXM 5.1 data is returned to the client.

The topic of providing portrayal capabilities in the Data Broker is discussed in more detail in Chapter 8.

7.4.2. Use case 2: Mediation and Brokering

The second use case focuses on the ability to support mediation and brokering of semantically equivalent data formats. This enables users to gather data in a single format from the Data Broker, while the data source formats can be different. The first figure below illustrates this with AIXM 5.1 as requested exchange format; the next figure illustrates this for AMXM 2.0.

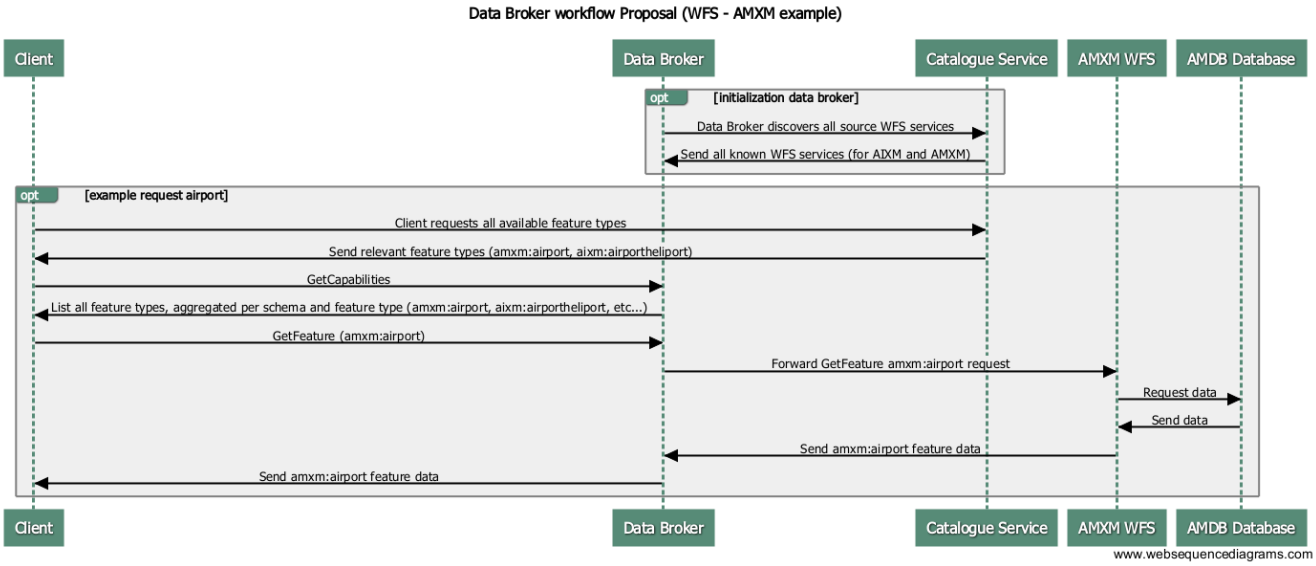

Figure 3. Request AMXM data as feature data, using WFS

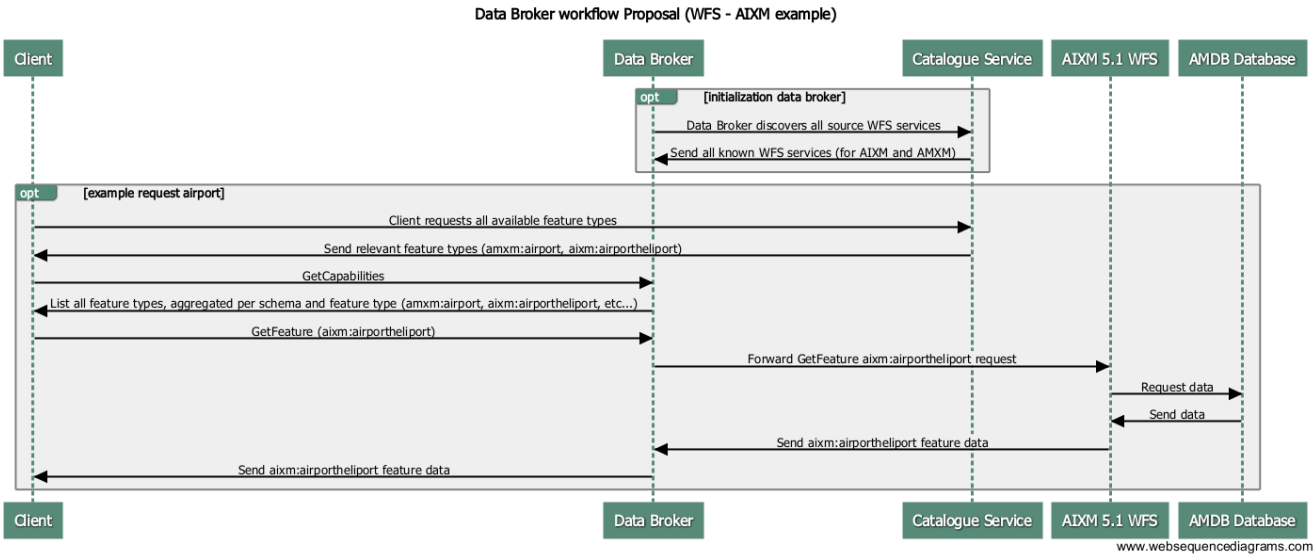

Figure 4. Request AIXM data as feature data, using WFS

From a high-level perspective, the following steps take place for both use cases:

1. As initialization for the Data Broker, it will connect to the CSW to retrieve all possible WFS and WMS endpoints
2. As initialization for the client, it will connect to the CSW to retrieve the Data Broker endpoint
3. The client retrieves the WFS capabilities document of the Data Broker. In this process it will get an overview of all the feature types that are aggregated from AIXM and AMXM data sources.
4. The Client requests airport feature type (in either AMXM or AIXM form)
5. The Data Broker retrieves the data from one of more of its aggregated WFS services.
6. The Data Broker enriches the data with lineage information and returns it to the Client.

The topic of brokering semantically equivalent data formats is further discussed in Chapter 9.

7.4.3. Use case 3: Portrayal

A third use case relates to the portrayal use case: next to offering portrayal capabilities, a Data Broker could also include portrayal information in its WFS response. This use case is illustrated in the figure below.

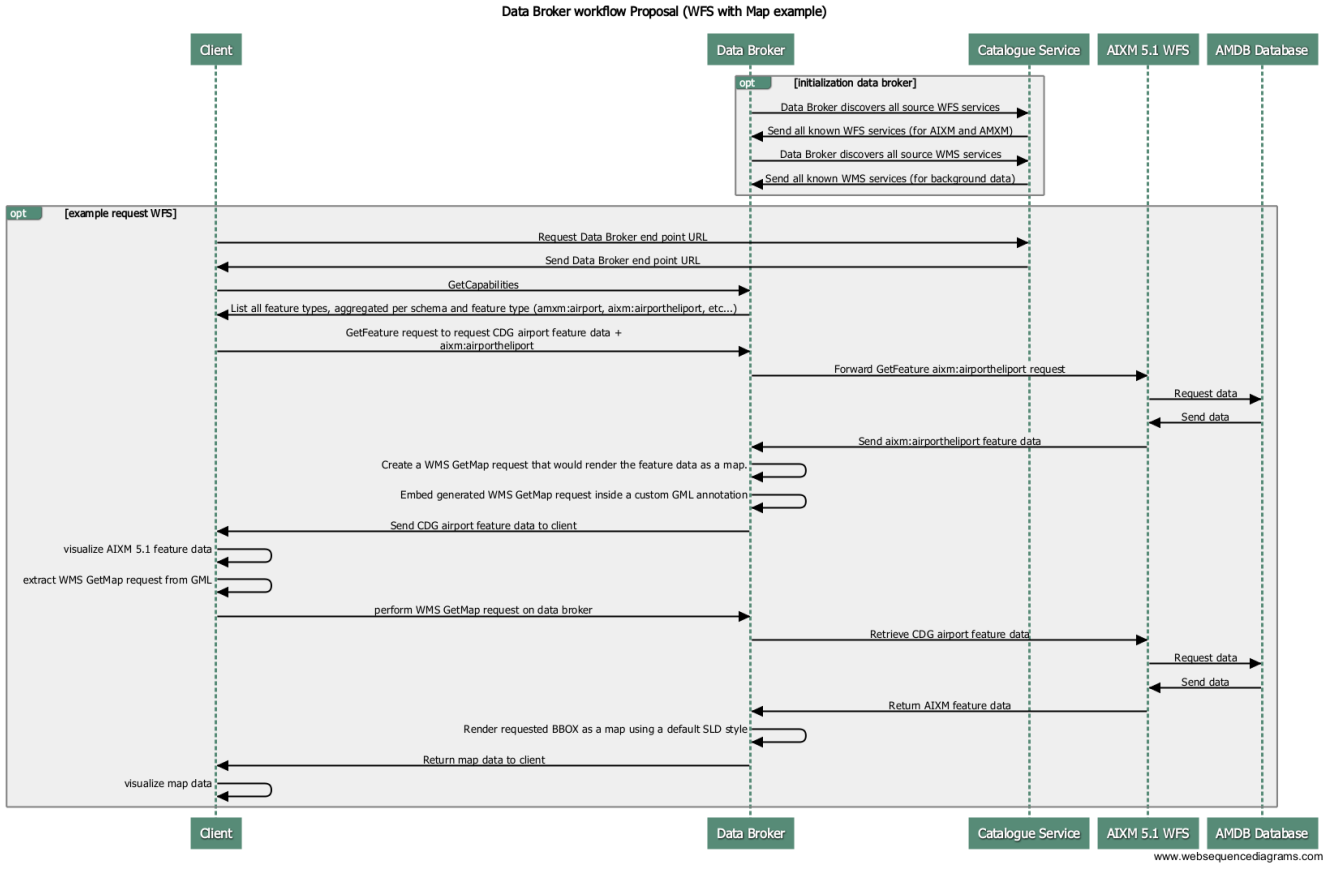

Figure 5. Request feature data as map, with link to the original feature data

On a high level, the following steps take place:

1. As initialization of the Data Broker, it will connect to the CSW to retrieve all possible WFS sources.
2. As initialization for the client, it will connect to the CSW and retrieve the Data Broker endpoint.

3. The client retrieves the WFS capabilities document. In this process it will get an overview of all the feature types that are aggregated from AIXM and AMXM data sources
4. The client requests an airport as feature data
5. The data broker requests the feature data from one or more of its aggregated WFS services.
6. The data broker enriches the feature data with an annotation that can be used for portrayal.
7. The data broker returns the result to the client.
8. The client uses the annotation to perform portrayal using the WMS service of the Data Broker.
9. The Data Broker retrieves the feature data from the source WFS service
10. The Data Broker returns rendered map data to the client.

In step 6, the data broker enriches feature data with additional information on how to portray the resulting WFS service. This is usually in the form of an URL, where a GetMap request is performed back on the Data Broker. An overview of how this can be achieved in practices is part of Chapter 10.

7.4.4. Use case 4: Dynamic Discovery

The last use case is the dynamic discovery of data sources within the Data Broker. This topic is discussed in more detail in Chapter 11.

Chapter 8. Providing portrayal capabilities

To support Data Broker clients that do not include an OGC WFS connector and / or support for GML-based data exchange standards, it can be useful to equip the Data Broker with an OGC WMS protocol. This is ideally suited to support rendering use cases. Additionally, clients can rely on the WMS GetFeatureInfo request to retrieve more information about a particular feature. The first section discusses the portrayal architecture in more detail. This is followed by a discussion about defining the WMS capabilities.

8.1. Portrayal architecture

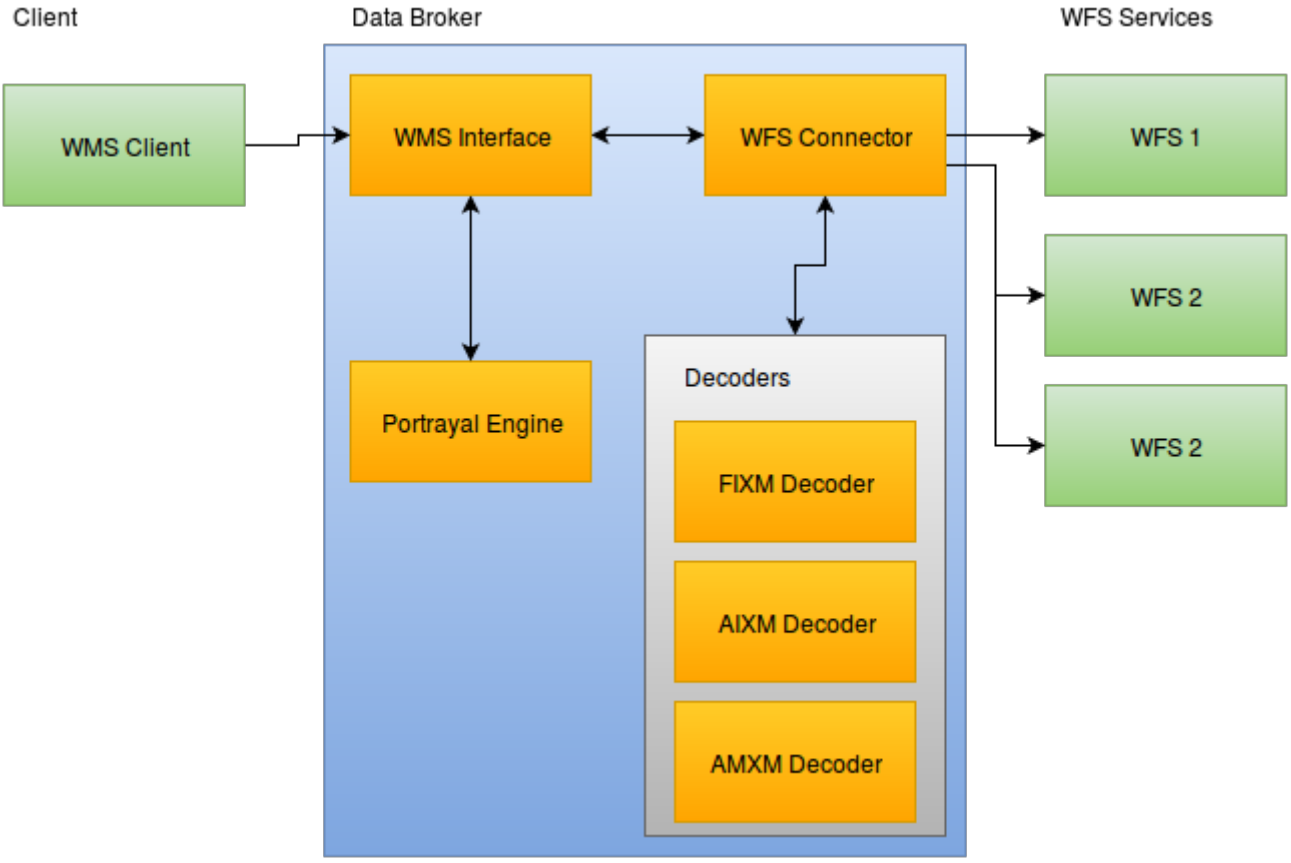

Figure 6. High-level portrayal architecture components and flow

8.2. Defining the set of WMS layers

An essential part of adding an OGC WMS to the Data Broker is the definition of the available WMS layers, listed in the capabilities. The Data Broker's WFS defines a list of feature types representing the available feature data. This list can be mapped one-to-one on a list of WMS layers. By definition, the resulting list is grouped in one root WMS layer.

As an example, the following feature type is returned from the Data Broker when listing WFS feature types:

```
 <wfs:FeatureType>
   <wfs:Name>aixm:AirportHeliport_aixm</wfs:Name>
     <wfs:Title>AirportHeliport</wfs:Title>
     <wfs:Abstract>AirportHeliport</wfs:Abstract>
     <wfs:DefaultSRS>urn:ogc:def:crs:EPSG::4326</wfs:DefaultSRS>
     <wfs:Operations>
       <wfs:Operation>Query</wfs:Operation>
     </wfs:Operations>
     <wfs:OutputFormats>
       <wfs:Format>GML32</wfs:Format>
       <wfs:Format>text/xml</wfs:Format>
       <wfs:Format>text/xml; subtype=gml/3.2.1</wfs:Format>
       <wfs:Format>application/gml+xml; version=3.2</wfs:Format>
       <wfs:Format>AIXM51</wfs:Format>
     </wfs:OutputFormats>
     <ows:WGS84BoundingBox>
       <ows:LowerCorner>-180.0 -90.0</ows:LowerCorner>
       <ows:UpperCorner>180.0 90.0</ows:UpperCorner>
     </ows:WGS84BoundingBox>
   </wfs:FeatureType>
```

Then the resulting WMS Layer will look like this:

```
 <Layer queryable="0">
   <Name>AirportHeliport_aixm</Name>
   <Title>AirportHeliport_aixm</Title>
   <Abstract>WMS layer containing AirportHeliport data served by the Data Broker
WFS.</Abstract>
   <KeywordList>
   <Keyword>M-Click AIXM</Keyword>
   <Keyword>Snowflake AIXM</Keyword>
   </KeywordList>
   <EX_GeographicBoundingBox>
   <westBoundLongitude>-180.0</westBoundLongitude>
   <eastBoundLongitude>180.0</eastBoundLongitude>
   <southBoundLatitude>-90.0</southBoundLatitude>
   <northBoundLatitude>90.0</northBoundLatitude>
   </EX_GeographicBoundingBox>
   <BoundingBox CRS="CRS:84" minx="-180.0" miny="-90.0" maxx="180.0" maxy="90.0"/>
 </Layer>
```

8.3. Lineage

An important aspect of the Data Broker's WFS is the integration of lineage information in the feature data to identify a feature's data source. This is obviously no longer possible in a WMS context, since the result is bitmap. To still inform WMS clients about the underlying data source, we can include relevant metadata in the WMS layer definitions in the capabilities.

8.3.1. Attribution

A WMS layer has an optional Attribution property, which indicates the provider of a WMS layer. This can include the provider's URL, a descriptive title string and even a logo image URL. The Data Broker could rely on this property to list the WFS data sources providing the feature data rendered by a WMS layer.

```
<Layer queryable="0">
  <Name>AirportHeliport_aixm</Name>
  <Title>AirportHeliport_aixm</Title>
  <Abstract>WMS layer containing AirportHeliport data served by the Data Broker
WFS.</Abstract>
  <KeywordList>
    <Keyword>M-Click AIXM</Keyword>
    <Keyword>Snowflake AIXM</Keyword>
  </KeywordList>
  <EX_GeographicBoundingBox>
    <westBoundLongitude>-180.0</westBoundLongitude>
    <eastBoundLongitude>180.0</eastBoundLongitude>
    <southBoundLatitude>-90.0</southBoundLatitude>
    <northBoundLatitude>90.0</northBoundLatitude>
  </EX_GeographicBoundingBox>
  <BoundingBox CRS="CRS:84" minx="-180.0" miny="-90.0" maxx="180.0" maxy="90.0"/>
<Attribution>
    <Title>M-Click</Title>
    <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
xlink:href="https://demo.m-click.aero/services/wfs-tb12aixm" />
</Attribution>
</Layer>
```

According to the WMS specification, clients may choose to display attribution on the map. For clients that support this feature, it would mean that the attribution would be clearly visible while viewing the map data.

One disadvantage of using attribution is that it does not allow for a hierarchy of lineage information. If the source WFS service is part of a chain, then it would not be possible to trace the data back to the originator. The relationship between attribution and map is also lost, as there is no capability to express anything other than a title and a link.

8.3.2. MetadataURL

A WMS layer has an optional MetadataURL property, which can link to a metadata element offering detailed, standardized metadata about the data underneath the layer. The Data Broker could rely on this property to link it to an ISO 19115-based metadata description, similar as how it was used in the WFS features.

```
<Layer queryable="0">
  <Name>AirportHeliport_aixm</Name>
  <Title>AirportHeliport_aixm</Title>
  <Abstract>WMS layer containing AirportHeliport data served by the Data Broker
WFS.</Abstract>
  <KeywordList>
    <Keyword>M-Click AIXM</Keyword>
    <Keyword>Snowflake AIXM</Keyword>
  </KeywordList>
  <EX_GeographicBoundingBox>
    <westBoundLongitude>-180.0</westBoundLongitude>
    <eastBoundLongitude>180.0</eastBoundLongitude>
    <southBoundLatitude>-90.0</southBoundLatitude>
    <northBoundLatitude>90.0</northBoundLatitude>
  </EX_GeographicBoundingBox>
  <BoundingBox CRS="CRS:84" minx="-180.0" miny="-90.0" maxx="180.0" maxy="90.0"/>
  <MetadataURL type="simple">
    <Name>Metadata URL</Name>
    <Format>application/iso19115+xml</Format>
    <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
                    xlink:type="simple"

xlink:href="http://demo.luciad.com:8080/OgcTestbedServices/broker?service=WMS&request=
DescribeMetadata&version=1.3.0&layers=AirportHeliport_aixm"/>
  </MetadataURL>
</Layer>
```

In the example above, the capabilities document of the Data Broker WMS is enriched with a MetadataURL property, which links to an online resource. This online resource is a URL generated by the Data Broker that links to the ISO 19115-based XML file containing lineage information about the layer itself. For our use case, we introduced a new request type "DescribeMetadata" for this purpose. But this could also be a link to a static file. In practice it is easier to create the ISO 19115 dynamically on the fly, based on the response of the WFS service.

An ISO 19115 XML document with lineage information would look something like this:

```
  ...
  <gmd:lineage>
    <gmd:LI_Lineage>
      <gmd:source>
        <gmd:LI_Source>
          <gmd:description>
            <gco:CharacterString>https://demo.m-click.aero/services/wfs-
tb12aixm</gco:CharacterString>
          </gmd:description>
          <gmd:sourceCitation>
            <gmd:CI_Citation>
              <gmd:title>
                <gco:CharacterString>M-Click</gco:CharacterString>
              </gmd:title>
              <gmd:date>
                <gmd:CI_Date>
                  <gmd:date>
                    <gco:DateTime>2016-07-26T16:48:45.134+02:00</gco:DateTime>
                  </gmd:date>
                  <gmd:dateType>
                    <gmd:CI_DateTypeCode
codeList="iso/19139/resources/gmxCodelists.xml#CI_DateTypeCode"
codeListValue="creation">creation</gmd:CI_DateTypeCode>
                  </gmd:dateType>
                </gmd:CI_Date>
              </gmd:date>
            </gmd:CI_Citation>
          </gmd:sourceCitation>
          <gmd:sourceStep>
          ....
```

The advantage of using MetadataURL is that each WMS layer can have its own lineage information inside of it. Another advantage is that the lineage information can express multiple levels of lineage. This means that in a hierarchy with multiple data brokers, a client will be able to trace the origin all the way back to the original source of the data.

8.4. GetFeatureInfo

To enable WMS clients accessing more information about a visualized feature, the Data Broker's WMS can offer GetFeatureInfo support. The WMS standard does not define any standardized exchange format, so we can define a few possibilities that match with the Data Broker's use case:

- Use of the native feature data format, i.e. AIXM 5.1 or AMXM 2.0. This allows a WMS client to get the actual feature, as provided by the WFS.
- Use of GeoJSON. Being an open and simple format to represent geographical features, it is ideally suited to support web-based clients.

8.5. Styling

A standard WMS uses one or more predefined styles for each offered layer, from which a client can choose. A WMS with support for the Styled Layer Descriptor (SLD) Profile extends this with the ability to include an SLD in a request, enabling users to customize the styling of a WMS layer.

Within a Data Broker, both capabilities are useful:

- A predefined style enables lightweight clients without SLD support to easily retrieve maps. A logical choice for the predefined style is the application of ICAO Annex 4 aeronautical charting guidelines [9]. This ICAO document includes symbols, line styles and label guidelines for a wide range of aeronautical entities.
  - This can be extended with multiple predefined styles using different color schemes - e.g., based on the time of the day.
- With SLD support, it is possible for a client to customize the styling to very specific needs - e.g., adding a safety zone around aeronautical objects such as obstacles.

8.6. Filtering

In contrast to the styling options, data filtering options are limited. In a standard WMS, data cannot be filtered, except for a geographical area of interest that is supplied with a map request. To filter data based on its properties - for instance, to filter all airports that are located in a given ICAO region - an SLD-enabled WMS is required: next to a styling definition, an SLD can include an OGC Filter to filter the data during the rendering process. One drawback of this approach is the fact that the filter is an integral part of the styling definition: it cannot be used independently of a customized style, so users will always need to define a style next to the filter - which might not be desirable, if the default, predefined style is preferred.

For WMS layers serving rendered vector data, such as the Data Broker's WMS, it would be useful to let clients include filters in map requests to filter the layer's data content. One practical example in the geospatial community can be found in GeoServer, which supports filter parameters in a map request - expressed as an OGC Filter or using ECQL.

In case the offered filtering options can be limited - such as only allowing to filter on ICAO region - it could also be an option to rely on WMS dimensions. WMS dimensions make it possible to parameterize a layer with one or more parameters. Typical examples are altitude and time, but it is equally possible to define a custom dimension - such as an ICAO region. Although this approach limits the filtering possibilities (you can only filter based on what is offered by the layer's dimension(s), it is part of the OGC WMS standard - hence an interoperable solution.

Chapter 9. Brokering semantically equivalent data: AIXM 5.1 and AMXM 2.0

The Data Broker developed within Testbed 11 focused on a WFS brokering capability using AIXM 5.1 as exchange format. Within Testbed 12, this is extended with a second, semantically equivalent exchange format, AMXM 2.0. Although the brokering capability is essentially format independent, this chapter reviews the existing architecture in the light of this extension. An overview is given on several brokering aspects specifically related to AMXM 2.0. This is followed by a discussion about leveraging the semantic equivalence of AIXM 5.1 and AMXM 2.0 and exploiting possible use cases in a Data Broker context.

9.1. AMXM 2.0 & feature collections

An AIXM 5.1 data set essentially consists of a set of features, contained in a feature collection. In a stand-alone context, this feature collection is an AIXM BasicMessage. Within a WFS context, this feature collection is by definition a WFS FeatureCollection. In AMXM 2.0, there isn't a feature collection element. Features can still be grouped, but within another, core feature: the AerodromeMappingDatabase. In theory, this AerodromeMappingDatabase could be added as single feature in a WFS context, ending up with a WFS FeatureCollection containing a single AerodromeMappingDatabase feature containing all available airport layout features (Aprons, Taxiways …). Aligned with the standard, the WFS only knows about the AerodromeMappingDatabase features - the underlying features are considered properties - so a response will either be empty or contain this AerodromeMappingDatabase feature, with all its subfeatures. This is not particularly useful, since it avoids users to be able to do a fine-grained selection on airport layout features. The Data Broker implementation follows a different approach, in which the AerodromeMappingDatabase element is left out and in which the contained airport layout features are directly put under a WFS FeatureCollection.

9.2. AMXM 2.0 & Lineage

An important aspect of the Data Broker is to store information about the source of the data, to inform clients about the original provider. This is also known as lineage information. A common and standardized way to represent such information is the ISO 19115 metadata model. Within AIXM 5.1 [3], ISO 19115 metadata can be used throughout the data model:

- At a feature collection level
- At a feature level
- At a feature's timeslices level

Within the Data Broker, the need is primarily at the feature level. Looking at AMXM 2.0, we have slightly different metadata provisions. ISO 19115 metadata can be used, but only on the AerodromeMappingDatabase feature level [2]. As discussed in the previous section, we chose to leave this feature out to better align AMXM and WFS feature provision. Consequently, this has as drawback that we can no longer store metadata on a feature level.

We could address this by replacing the WFS FeatureCollection with an AMXM AerodromeMappingDatabase and use this to include Lineage information, but this would be a deviation of the WFS standard - which dictates that the output should be encompassed in a WFS FeatureCollection. Additionally, including it on an AerdromeMappingDatabase level would make it global for all encompassed AMXM features, while these may be provided by different WFS sources used by the Data Broker.

Based on the above analysis, a recommendation for the AMXM standard is to add an ISO metadata property on all AMXM features - similar to AIXM. This gives users the ability to store fine-grained metadata on an individual feature - such as lineage information.

9.3. AMXM 2.0 & Conflation

A Data Broker is responsible to conflate features, to ensure that there are no duplicates in the resulting Broker's response. The preferred way to do this is to rely on unique identifier information available in the features. Similar to AIXM 5.1, AMXM 2.0 enables the use of a unique identifier, so this requirement is fulfilled. The Data Broker implementation in Testbed 12 shows that conflation equally works for both AIXM 5.1 and AMXM 2.0.

9.4. Mediating between AIXM 5.1 and AMXM 2.0

With the Data Broker able to support two exchange formats that share a lot of concepts (airport layout entities) and underlying technology (GML 3.2), we can look at leveraging these similarities. One use case is to support data mediation capabilities and enable the broker to offer heterogeneous data formats into a single format to the client.

The Data Broker's implementation of Testbed 12 supports transforming AMXM 2.0 to AIXM 5.1 data, on-the-fly during the brokering process. To include this capability, the existing Data Broker's processing pipeline from Testbed 11 has been extended with a mediation component, as illustrated in the figure below.

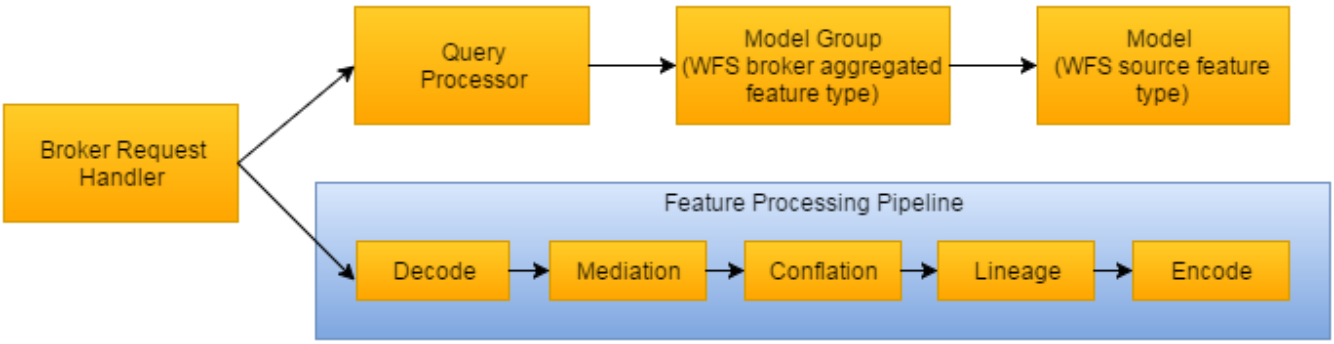

Figure 7. The Data Broker's Feature Processing Pipeline, extended with a Mediation component

The integration of data mediation does impact some of the Data Broker's responsibilities:

- **Conflation**: To support conflation on a combination of mediated and native features, it is important to be able to identify features in the exact same way. With the Data Broker relying on a feature's unique identifier property, it is therefore required that data providers use the same id for aeronautical entities, regardless of the format encoding; for instance, an apron encoded in both AIXM and AMXM should use the same unique id. If this requirement cannot be met, the

Data Broker needs to use enhanced conflation techniques to identify duplicate features, based on properties other than identifiers - for instance, using properties such as the name, ICAO code, and possibly even the geometry.

- **Lineage**: Because the mediation step is a processing step affecting a feature, it is important to record this in the feature's lineage metadata. When mediating from AMXM to AIXM, this is possible by means of the ISO metadata property available on AIXM features. This gets more difficult when mediating from AIXM to AMXM, because of the lack of an (ISO) metadata property on an AMXM feature - hence the earlier recommendation to include an ISO metadata property on each AMXM feature.

Additionally, the Data Broker should offer the mediation capability to its clients in an easy-to-use and interoperable way, hiding the details related to heterogeneous data aggregation and mediation (apart from their description in the metadata). One logical way to do this is to define a feature type per aeronautical entity, such as AirportHeliport, Airspace, ApronElement, … and let the client choose the resulting format by means of the OUTPUTFORMAT request parameter. Each feature type might be an aggregation of multiple data sources using heterogeneous formats; using the OUTPUTFORMAT request parameter and the Data Broker's mediation capability, the requested data is seamlessly collected and sent in a single format to the client.

Chapter 10. Including portrayal information in a feature response

To ease rendering of feature data served by the Data Broker, it could be helpful to include a portrayal link in its WFS responses. This portrayal link could refer to its built-in WMS described previously, or to an external FPS capable of rendering remote (linked or embedded) data. Section 1 discusses how a portrayal link can look like. Another main discussion topic is how to include this portrayal link in a WFS response. Section 2 discusses possible approaches in more detail.

10.1. Defining a portrayal link

10.1.1. Using WMS named layer(s)

To support the rendering of data provided by the Data Broker, it can be equipped with an OGC WMS, offering WMS named layers corresponding to the WFS feature types - as discussed in a previous chapter. Thus, an easy approach for the portrayal link in a WFS response could be to refer to the WMS named layer(s) corresponding to the requested WFS feature type(s). Apart from the WMS layer name(s), the link could include the area of interest (BBOX) parameter potentially used in the initial Data Broker's WFS GetFeature request.

10.1.2. Using WMS user-defined layer(s)

While the use of WMS named layers offers an easy way to access rendered versions of the Broker's data, it does not offer the flexibility to get a rendered version of the exact same data included in a WFS response. Although a BBOX spatial filter can be taken into account, a WFS GetFeature request can also include other parameters that influence the data content – such as COUNT or in general any OGC Filter. This cannot be used as such in a WMS GetMap request.

To support this use case, we need to look at an FPS, i.e. a WMS supporting the SLD profile and its ability to customize the content of a layer by means of user-defined layers. With this approach, the portrayal link could refer to the Data Broker's WFS response content. This can be done in multiple ways:

By embedding the response content

An SLD user-defined layer allows using inline features, i.e. features embedded in the SLD. This could be used to include the features of the Broker's response. The main disadvantage of this approach is the size of the resulting SLD. Because of this, only XML POST request encodings can be used. A simple KVP request is thus no longer possible. Additionally, it poses a conflict with AIXM 5.1 and AMXM 2.0 from a standard's interoperability perspective. Embedding features is only possible if they are based upon GML 3.1., according to the SLD 1.1 XML Schema [4]:

```
<xsd:schema ···xmlns:gml="http://www.opengis.net/gml" ···>
<xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
···

<xsd:element name="InlineFeature">
<xsd:complexType>
<xsd:sequence>
<xsd:element ref="gml:FeatureCollection" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
```

However, both AIXM 5.1 and AMXM 2.0 are based upon GML 3.2. Consequently, we cannot use AIXM / AMXM features in inside an SLD. Finally, it also results in data duplication since the feature would appear both in the WFS response's feature collection and in the SLD.

By linking to the WFS

Another approach is to let the user-defined layer link to the WFS feature type(s), including the use of possible filter parameters. Compared to the previous approach, it avoids the GML 3.1.1 compatibility issue and it is much smaller in size.

Example:

```
<sld:RemoteOWS>
  <sld:Service>WFS</sld:Service>
  <se:OnlineResource xlink:href="
http://demo.luciad.com:8080/OgcTestbedServices/broker?REQUEST=GetFeature&SERVICE=WFS&V
ERSION=2.0.0&TYPENAMES=AirportHeliport&COUNT=1" xlink:type="simple"/>
</sld:RemoteOWS>
```

A complete example of a resulting SLD can be found here.

This approach avoids the GML 3.1 (SLD) and GML 3.2 (AIXM / AMXM) conflict and it avoids feature duplication. However, it can still be that the request URL gets too long in HTTP GET form, so there might still be a need to switch to a POST request – if this is possible inside processing instruction or ISO 19115 online resource.

Defining the style

When SLD user-defined layers are used, it is by definition also required to include user-defined styling. After all, an FPS is not in general not aware of the user-defined data content, hence it does not know how to style it. To resolve this, a user-defined layer comes with a user-defined style, including an OGC SE Feature Type Style. This Feature Type Style defines the styling information needed to render the data. The styling information can be defined in two ways:

- Through embedded styling information

- Through a remote style link

Both options are equally valid; the preferred choice depends on the use case:

- In case the user is interested in reviewing and possibly changing the style, the first option is the most appropriate.
- In case the user is only interested in the default style, the second option can be chosen, since it helps to reduce the request size.

10.2. Embedding portrayal links in a feature response

10.2.1. As XML processing instruction

An XML processing instruction is an XML node type intended to carry instructions to an application that reads the XML document. It may occur anywhere in the XML document, although, if present, it can often be found before the XML document's root element. The most common use of a processing instruction is a link to an XSLT or CSS stylesheet, supporting the rendering of the XML data.

```
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
```

This use case is very similar to the Data Broker's portrayal link use case, since the latter is also defined to support the rendering of the Data Broker's XML response.

Example:

```
<?xml-portrayal href="http://WMS" type="image/png"?>

<FeatureCollection xmlns="http://www.opengis.net/wfs/2.0">

<!-- feature members -->

</FeatureCollection>
```

10.2.2. Within the feature data

An alternative to an XML processing instruction is to embed a portrayal link within the feature data itself. To make this possible, the feature data's XML Schema should enable the possibility to include such metadata. The following paragraphs discuss this approach in more detail.

Within the Data Broker, two feature data formats are being looked at: AIXM 5.1 and AMXM 2.0. Both support the use of ISO 19115 metadata, a vast metadata model providing fine-grained possibilities to describe data. This includes the possibility to add portrayal link information.

Example:

```
<gmd:MD_Metadata>
  <gmd:distributionInfo>
    <gmd:MD_Distribution>
      <gmd:distributionFormat>
        <gmd:MD_Format>
          <gmd:name>
            <gco:CharacterString>image/png</gco:CharacterString>
          </gmd:name>
        </gmd:MD_Format>
      </gmd:distributionFormat>
      <gmd:transferOptions>
        <gmd:MD_DigitalTransferOptions>
          <gmd:onLine>
            <gmd:CI_OnlineResource>
              <gmd:linkage>
<gmd:URL>http://demo.luciad.com:8080/OgcTestbedServices/fps?SERVICE=WMS&amp;REQUEST=Ge
tMap&amp;VERSION=1.3.0&amp;LAYERS=DesignatedPoint&amp;FORMAT=image/png&amp;BBOX=0,0,10
,10&amp;CRS=CRS:84&amp;WIDTH=800&amp;HEIGHT=600&amp;TRANSPARENT=TRUE</gmd:URL>
              </gmd:linkage>
              <gmd:protocol>
                <gco:CharacterString>WMS</gco:CharacterString>
              </gmd:protocol>
            </gmd:CI_OnlineResource>
          </gmd:onLine>
        </gmd:MD_DigitalTransferOptions>
      </gmd:transferOptions>
    </gmd:MD_Distribution>
  </gmd:distributionInfo>
</gmd:MD_Metadata>
```

10.2.3. At a feature collection level

The feature collection refers to the group of features present in a dataset, and is typically represented by the XML root element encompassing the features. Because it is global for contained features, it is thus an ideal place to store cross-feature information – such as a portrayal link for the combined features. As discussed above, this does require a property entry on the feature collection element to include such metadata. Strictly speaking, a GML-based WFS response should always use the WFS feature collection element [5]. In the current WFS 2.0 XML Schema, this feature collection element does not include any metadata entries other than bounding box information.

However, a WFS can support alternative output formats; hence, we could look at relaxing the WFS feature collection element requirement and investigating the use of alternative feature collection elements. Within the Data Broker, two formats are being looked at: AIXM 5.1 and AMXM 2.0. In case of AIXM 5.1, we have the AIXMBasicMessage element, which is a feature collection element extending from GML's abstract feature collection element – similar to a WFS feature collection element. In case of AMXM 2.0, we have the AerodromeMappingDatabase element, which is strictly speaking not a feature collection element but a feature extending from GML's abstract feature element. However, AMXM does use it to group a set of features, so it conceptually matches with our

requirement.

Both AIXMBasicMessage and AerodromeMappingDatabase include an entry to ISO 19115 metadata, hence both elements could be used to include the portrayal link.

10.2.4. At a feature level

To avoid changing the WFS feature collection element, we could also look at storing the portrayal link information on a feature level. In case of AIXM 5.1, ISO 19115 metadata can be used on a feature level. In case of AMXM 2.O, no similar metadata possibilities can be found on a feature level. Additionally, an important question for this approach is the feature choice: which feature should include the information? A few possibilities:

- Include it with each feature
- Include it with a core feature
- Include it in the first feature
- Use a specific feature to include it.

Chapter 11. Discovery of data sources through CSW

The OGC Catalogue Service for the Web (CSW) is a standard for exposing a catalogue of geospatial records. Common examples are records about data sets and web services. One potential use case of this is discovery of data, based on search terms that exploit the record's metadata. This discovery capability can be leveraged by the Data Broker. Within Testbed 11, the Data Broker was preconfigured with a set of WFS data sources upon startup. By integrating the Data Broker with a CSW, it can automatically discover data sources, without any manual preconfiguration.

This chapter describes in more detail the interaction between the Data Broker and the CSW. The following sections discuss how the discovery through CSW can be done in practice. This is followed by a conclusion about the CSW integration experiments performed within Testbed 12.

11.1. Discovery from a data point of view

As a CSW can expose metadata records about data sets, we can use it in the Data Broker to search for data sets of interest and consequently determine what the Data Broker can serve. The following example request searches for data sets based on a geographic region.

```
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
  xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
  xmlns:ogc="http://www.opengis.net/ogc"
  service="CSW"
  version="2.0.2"
  maxRecords="100"
  resultType="results">
  <Query typeNames="wrs:ExtrinsicObject">
    <ElementSetName>full</ElementSetName>
    <Constraint version="1.1.0">
      <ogc:Filter>
        <ogc:And>
          <ogc:PropertyIsEqualTo>
            <ogc:PropertyName>@objectType</ogc:PropertyName>
            <ogc:Literal>urn:ogc:def:ebRIM-ObjectType:OGC:Dataset</ogc:Literal>
          </ogc:PropertyIsEqualTo>
          <ogc:BBOX>
<ogc:PropertyName>rim:Slot[@name='http://purl.org/dc/terms/spatial']/wrs:ValueList/wrs
:AnyValue</ogc:PropertyName>
            <Envelope xmlns="http://www.opengis.net/gml"
srsName="urn:ogc:def:crs:EPSG::4326">
              <lowerCorner>49 -124</lowerCorner>
              <upperCorner>50 -123</upperCorner>
            </Envelope>
          </ogc:BBOX>
        </ogc:And>
      </ogc:Filter>
    </Constraint>
  </Query>
</GetRecords>
```

The resulting response includes a set of extrinsic objects, identifying data sets that correspond to the specified search criteria:

```
<?xml version="1.0"?>
<csw:GetRecordsResponse xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:xlink="http://www.w3.org/1999/xlink" version="2.0.2">
    <csw:SearchStatus timestamp="2016-10-06T11:27:23Z"/>
    <csw:SearchResults elementSet="full" numberOfRecordsMatched="30"
numberOfRecordsReturned="30" nextRecord="0">
        ...
        <wrs:ExtrinsicObject id="urn:uuid:86ec93ce-ab82-4db6-a77e-5e0316473a56"
lid="urn:uuid:86ec93ce-ab82-4db6-a77e-5e0316473a56" objectType="urn:ogc:def:ebRIM-
ObjectType:OGC:Dataset" status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
mimeType="application/xml">
            <rim:Slot name="http://purl.org/dc/terms/spatial"
slotType="urn:ogc:def:dataType:ISO-19107:2003:GM_Envelope">
                <wrs:ValueList>
                    <wrs:AnyValue>
                        <gml:Envelope xmlns:gml="http://www.opengis.net/gml/3.2"
srsName="urn:ogc:def:crs:OGC:1.3:CRS84">
                            <gml:lowerCorner>-180.0 -90.0</gml:lowerCorner>
                            <gml:upperCorner>-30.0 90.0</gml:upperCorner>
                        </gml:Envelope>
                    </wrs:AnyValue>
                </wrs:ValueList>
            </rim:Slot>
            <rim:VersionInfo versionName="20160902T192349Z"/>
            <rim:ContentVersionInfo versionName="20160902T192349Z"/>
            <wrs:repositoryItemRef xlink:type="simple"
xlink:href="http://ows.galdosinc.com:80/indicio/query?request=GetRepositoryItem&amp;se
rvice=CSW-ebRIM&amp;id=urn:uuid:86ec93ce-ab82-4db6-a77e-5e0316473a56"/>
        </wrs:ExtrinsicObject>
        ...
    </csw:SearchResults>
</csw:GetRecordsResponse>
```

The extrinsic object includes a link to further identify the data set by means of an ISO 19115 data description, with information about the type of data, its geographic region and spatial reference and its provider.

This search capability enables a Data Broker to automatically find, aggregate and serve data sets based on user-defined data criteria.

11.2. Discovery from a web services point of view

Another approach for the Data Broker's CSW integration can be to search for web services. Because the Data Broker relies on the WFS, it can simply look for any web services adhering to this geospatial web service standard. Example request:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetRecords xmlns="http://www.opengis.net/cat/csw/2.0.2"
  xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
  xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
  xmlns:ogc="http://www.opengis.net/ogc"
  service="CSW"
  version="2.0.2"
  maxRecords="10"
  resultType="results">
  <rim:AdhocQuery id="urn:ogc:def:query:OGC-CSW-ebRIM::findServices">
    <rim:Slot name="serviceType">
      <rim:ValueList>
        <rim:Value>urn:ogc:def:serviceType:OGC::WFS</rim:Value>
      </rim:ValueList>
    </rim:Slot>
    <rim:Slot name="elementSetName">
      <rim:ValueList>
        <rim:Value>full</rim:Value>
      </rim:ValueList>
    </rim:Slot>
  </rim:AdhocQuery>
</GetRecords>
```

The resulting response includes a set of service objects, identifying web services that comply with the search criteria:

```
<?xml version="1.0"?>
<csw:GetRecordsResponse xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
xmlns:wrs="http://www.opengis.net/cat/wrs/1.0"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:xlink="http://www.w3.org/1999/xlink" version="2.0.2">
    <csw:SearchStatus timestamp="2016-10-06T11:40:52Z"/>
    <csw:SearchResults elementSet="full" numberOfRecordsMatched="5"
numberOfRecordsReturned="5" nextRecord="0">
        ...
        <rim:Service id="urn:uuid:a8de8ddc-23f1-458d-8dd5-d987a580443b"
lid="urn:uuid:a8de8ddc-23f1-458d-8dd5-d987a580443b"
objectType="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Service"
status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted">
            <rim:Slot name="http://purl.org/dc/terms/accessRights"
slotType="urn:oasis:names:tc:ebxml-regrep:DataType:String">
                <rim:ValueList>
                    <rim:Value>For OGC Testbed 12 demonstration purposes only. Not for
operational use.</rim:Value>
                </rim:ValueList>
            </rim:Slot>
            <rim:VersionInfo versionName="20160902T192336Z"/>
            <rim:Classification id="urn:uuid:1a3118f5-87a3-4ef0-a4d9-cf4f308dcb62"
lid="urn:uuid:1a3118f5-87a3-4ef0-a4d9-cf4f308dcb62"
objectType="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:Classification"
status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
classifiedObject="urn:uuid:a8de8ddc-23f1-458d-8dd5-d987a580443b"
classificationNode="urn:ogc:def:serviceType:OGC::WFS">
                <rim:VersionInfo versionName="20160902T192336Z"/>
            </rim:Classification>
            <rim:ServiceBinding id="urn:uuid:31bf8846-8653-4e72-a4f4-7b75b1556be3"
lid="urn:uuid:31bf8846-8653-4e72-a4f4-7b75b1556be3"
objectType="urn:oasis:names:tc:ebxml-regrep:ObjectType:RegistryObject:ServiceBinding"
status="urn:oasis:names:tc:ebxml-regrep:StatusType:Submitted"
service="urn:uuid:a8de8ddc-23f1-458d-8dd5-d987a580443b"
accessURI="https://demo.snowflakesoftware.com/go-publisher-wfs/TB12_AMXM_V20/wfs">
                <rim:Name>
                    <rim:LocalizedString xml:lang="en" charset="UTF-8"
value="GetCapabilities"/>
                </rim:Name>
                <rim:Description>
                    <rim:LocalizedString xml:lang="en" charset="UTF-8" value="Endpoint
for GET method."/>
                </rim:Description>
                <rim:VersionInfo versionName="20160902T192336Z"/>
            </rim:ServiceBinding>
        </rim:Service>
        ...
    </csw:SearchResults>
</csw:GetRecordsResponse>
```

Each service object includes service binding information, providing the necessary details to set up a connection with the service.

11.3. CWS integration approach in Testbed 12

Within Testbed 12, the CSW has been used by the Data Broker to discover web services, following the approach described in the previous section. The main benefit is avoiding manual web service configuration, which was the approach followed in Testbed 11. The discovery step is executed once, during the startup of the Data Broker. In an operational Data Broker component, this could be further extended with capabilities such as automatically & regularly scheduled discovery tasks to update the current list of web services.

Chapter 12. Lessons learned during integration

While integrating the Data Broker with other services, the following lessons were learned:

12.1. Metadata

For OGC Testbed 11, we introduced a data enrichment feature for the data broker. As data was being transformed or aggregated, this metadata was encoded into the resulting AIXM 5.1 result. For Testbed 12, some of the new formats do not have support for embedding ISO 19115 metadata in the response.

Both AMXM and FIXM do not currently support the ISO 19115 model to add rich metadata into the file itself. Having this would enable the Data Broker to add highly detailed lineage information on a feature level.

Alternatively, it would also have been desirable to be able to add ISO 19115 metadata into the a WFS FeatureCollection object. While this is less fine-grained than adding it directly on a feature level, it would allow for a more generic solutions to lineage metadata.

12.2. WMS Filtering

When describing a WMS service, the filtering capabilities you are presented with are limited to:

1. Spatial BBOX parameters
2. Dimension parameters

Outside of these parameters, additional filtering on the a named layer is not supported. For Testbed 12, we experimented with using the dimension parameter of WMS and using it as a filter on the GUFI of an aircraft. While this is not the intended use of the dimension parameter, it is a feasible option.

For more complex filter descriptions, it would be desirable to be able to construct more complex filtering parameters on a WMS named layer.

12.3. GML Compatibility

In OGC Testbed 12, we found that file formats based on GML 3.2 make it easier for a client to decode, if that client already supports GML 3.2 geometries.

This includes both AMXM and FIXM. The addition of GML simplified the process of decoding, parsing and visualizing the data.

Appendix A: Revision History

Table 2. Revision History

Date	Editor	Revision	Primary clauses modified	Descriptions
February 10 2016	0.1.0	Daniel Balog	all	Initial Version ER
March 1 2016	0.1.1	Daniel Balog	all	Added requirements clause, intro, scope and renamed the chapters.
March 2 2016	0.1.2	Daniel Balog	preface	Added Abstract, Business Value and value to Working Group.
March 3 2016	0.1.3	Robin Houtmeyers	preface	Improved Abstract and Business Values section.
June 20 2016	0.5	Robin Houtmeyers	all	Major update on draft content and structure
June 20 2016	0.6	Robin Houtmeyers	all	Added overview, bibliography and linking between chapters. Improved use case overview.
July 28 2016	0.7	Daniel Balog	all	Added section on relation to Workflow DWG. Reorganized some paragraphs and clarified content.

Date	Editor	Revision	Primary clauses modified	Descriptions
October 7 2016	0.8	Robin Houtmeyers	9,11	Made improvements based on feedback from sponsors. Added content to catalogue section.

Appendix B: Bibliography

1. Balog, D., Houtmeyers, R.: OGC Testbed 11 Data Broker Specifications Engineering Report, 15-028. OGC (2015).
2. RTCA, EUROCAE: AMXM 2.0 XML Schema, 'http://www.amxm.aero/schema/2.0.0/amxm.xsd'
3. Eurocontrol, FAA: AIXM 5.1 XML Schema, 'http://aixm.aero/document/aixm-51-xml-schema-xsd'
4. OGC :Styled Layer Descriptor 1.1 XML Schema, 'http://schemas.opengis.net/sld/1.1/StyledLayerDescriptor.xsd'
5. OGC: WFS 2.0 XML Schema, 'http://schemas.opengis.net/wfs/2.0/wfs.xsd'
6. Rieke M., Balaban A.: OGC Testbed 12 Asynchronous Messaging For Aviation, 16-017. OGC (2016)
7. ICAO: Annex 4 to the Convention on International Civil Aviation: Aeronautical Charts, Eleventh Edition (2009)