Open Geospatial Consortium

Date: 2012-01-25

Reference number of this document: OGC 11-072r2

Category: OGC® Public Engineering Report

Editors: Wenny Rahayu, Torab Torabi, Andrew Taylor-Harris, and Florian Puersch

# OGC® OWS-8 Aviation – Weather Information Exchange Model (WXXM) Engineering Report



**Warning**

*This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is 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.*

Document type: OpenGIS® Engineering Report
Document subtype: NA
Document stage: Approved for public release
Document language: English

## Preface

This document is a deliverable for the OGC Web Services 8 (OWS-8) testbed activity. OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver proven candidate standards or revisions to existing standards into OGC's Standards Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include test beds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

The OWS-8 sponsors are organizations seeking open standards for their interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommend to the sponsors that the content of the OWS-8 initiative be organized around the following threads:

* Observation Fusion

* Geosynchronization (Gsync)

* Cross-Community Interoperability (CCI)

* Aviation

More information about the OWS-8 testbed can be found at:

http://www.opengeospatial.org/standards/requests/74

OGC Document [11-139] “OWS-8 Summary Report” provides a summary of the OWS-8 testbed and is available for download:

https://portal.opengeospatial.org/files/?artifact_id=46176.

 

## License Agreement

# Contents

Page

 

# Figures



# Tables



 

 

# OGC® OWS-8 Aviation – WXXM Engineering Report

## 1 Introduction

### 1.1 Scope

This OGC™ document specifies the advancement of WXXM and Weather Concepts in the OWS-8 Aviation Thread. The focus is on investigating and demonstrating the applicability and suitability of WXXM in producing accurate, real-time aircraft weather radar data using OGC™ Web Coverage Services (WCS) to be used by meteorological applications and services supporting aviation. Such applications provide information which enhances safe and efficient tactical and pre-tactical decision making during planning and execution of a flight.

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 Inc. 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.

### 1.2 Document contributor contact points

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

| Name | Organization |
|---|---|
| **Wenny Rahayu (Editor)** | **La Trobe University** |
| **Torab Torabi (Editor)** | **La Trobe University** |
| **Florian Puersch** | **La Trobe University** |
| **Andrew Taylor-Harris** | **Airservices Australia** |
| **Peter Baumann** | **Jacobs University Bremen, Rasdaman GmbH** |
| **David Burggraf** | **Galdos** |
| **Richard Martell** | **Galdos** |
| **Russ Gillespie** | **Northrop Grumman Corporation** |
| **Joe Marshall** | **Northrop Grumman Corporation** |

 

### 1.3 Revision history

| Date | Release | Editor | Primary clauses modified | Description |
|---|---|---|---|---|
| **2011-05-04** | **0.1.0** | **WR TT** | **All** | **Definition of initial document outline** |
| **2011-06-23** | **0.2.0**<br>**OGC 11-072** | **All** | **All** | **Combination of all subclauses of clause 6** |
| **2011-08-26** | **0.3.0**<br>**OGC 11-072r1** | **AT FP** | **6.1, 6.2** | **Update to reflect input from other participants as well as own development** |
| **2011-09-30** | **1.0**<br>**OGC 11-072r2** | **All** | **6** | **Update to reflect input from other participants as well as own development and implementation** |

### 1.4 Future work

This document version is subject to formal review. Therefore, it is envisaged that it will be modified later when all contributions are collated.

### 1.5 Forward

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 Inc. 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.

## 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.

ISO 19115 (all parts), *Geographic information – Metadata*

ISO 19139, *Geographic information - Metadata - XML schema implementation*

ISO 19136:2007, *Geographic information — Geography Markup Language (GML)*

ISO 19123:2005, *Geographic information — Schema for coverage geometry and functions*

 

ISO/DIS 19142, *Geographic information — Web Feature Service*

ISO/DIS 19143, *Geographic information – Filter Encoding*

ISO/DIS 19156, *Geographic information – Observations and Measurements*

OGC 09-110r3, OGC® WCS (*Web Coverage Service) 2.0 Interface Standard - Core*

OGC 06-121r3, OGC® *Web Service Common Standard*

OGC 09-032, *OpenGIS® OWS-6-SWE Event Architecture Engineering Report*

OGC 09-050r1, *OpenGIS® OWS-6-AIM Engineering Report*

OGC 10-079, *OWS-7 Aviation Architecture Engineering Report*

OGC 10-131, *OWS-7 Aviation AIXM Assessment Report*

OGC 10-146r1. *OGC® GML 3.2.1 Application Schema for Coverages, version 1.0.0*

FAA/EATM. *WXXM 1.1 Primer*, draft version

Encoding Representative Weather Datasets

*https://portal.opengeospatial.org/twiki/pub/OWS8/AviationWeather/WXXS-Compliance.doc*

## 3 Terms and definitions

For the purposes of this report, the terms and definitions are defined in the OWS-6 Event Architecture Engineering Report (09-032) and the OWS-6 AIM Engineering Report (09-050r1).

## 4 Conventions

### 4.1 Abbreviated terms

AIM Aeronautical information Management

AIXM Aeronautical Information Exchange Model

API Application Programming Interface

EFB Electronic Flight Bag

| | |
|---|---|
| ER | Engineering Report |
| FE | Filter Encoding |
| FES | Filter Encoding Specification |
| FIR | Flight Information Region |
| FL | Flight Level |
| FPS | Feature Portrayal Service |
| GML | Geography Markup Language |
| HTTP | HyperText Transport Protocol |
| ICAO | International Civil Aviation Organization |
| ISO | International Standardization Organization |
| METAR | Metorological Aviation Report |
| NNEW | NextGen Network Enabled Weather |
| NOTAM | Notice to Airmen |
| OGC | Open Geospatial Consortium |
| OWS | OGC Web Services |
| OWS-8 | OWS Testbed phase 8 |
| O&M | Observation & Measurement |
| SIGMET | Significant Meteorological Information |
| SLD | Styled Layer Descriptor |
| SOA | Service Oriented Architecture |
| SWE | Sensor Web Enablement |
| TAF | Terminal Aerodrome Forecast |

UCAR University Corporation for Atmospheric Research

UDP User Datagram Protocol

UML Unified Modelling Language

UOM Unit of Measure

VA Volcanic Ash

WCS Web Coverage Service

WFS Web Feature Service

WFS-T Web Feature Service –Transactional

WMS Web Mapping Service

WXXM Weather Information Exchange Model

XML Extensible Markup Language

### 4.2 UML notation

Most diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3].

## 5 WXXM Assessment overview

### 5.1 Overview

WXXM, the Weather Data Model, is a UML-based structural definition for the exchange of weather information, featuring an Aviation-specific layer. It was designed by EUROCONTROL, in partnership with NNEW.

WXXM is not a piece of software, nor does it have any function on its own. It defines a common vocabulary for exchanging weather information between organizations, but it does not inherently provide any sort of functionality to facilitate this exchange. It is, fundamentally, a set of guidelines for how to think about weather data.

The Weather Data Model is, in fact, a set of three-tiered data models, only one part of which is actually called WXXM: Weather Exchange Model. Together, the three models provide conceptual, structural, and physical representations of weather data:

 

- The Weather Conceptual Model (WXCM) provides a high-level, implementation-independent look at how weather data concepts are connected.
- The Weather Exchange Model (WXXM) provides a more logical and structural (if still implementation-independent) perspective of the same data, in more complete detail — the interrelationships of every weather data concept are spelled out.
- The Weather Exchange Schema (WXXS) is a machine-generated, XML-formatted implementation of the Exchange Model — a "physical" code version of it.

As it has become a commonly used, "generic" term, WXXM will be used in the following sections of this document, irrespective of which part of the model is being referred to.

The different layers of the WXXM model are represented in Figure 1 below:

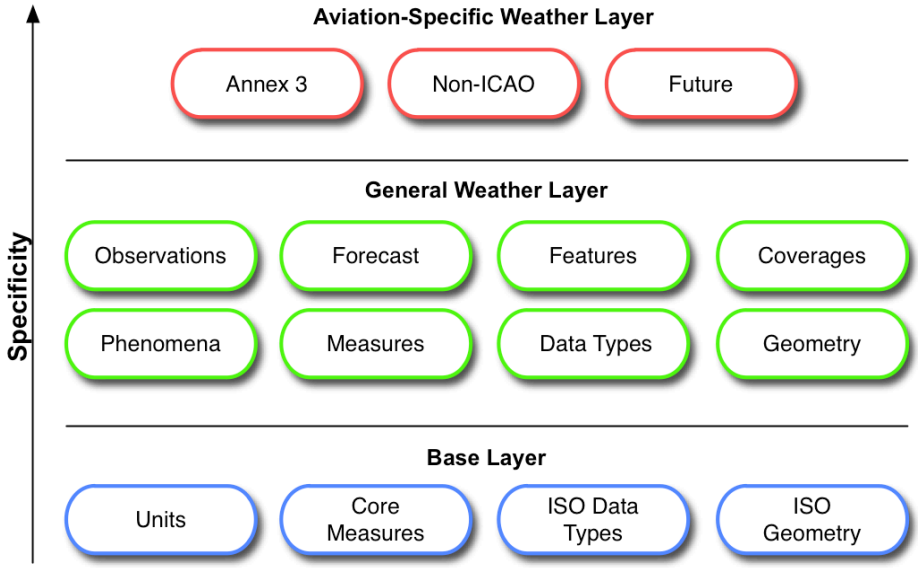


**Figure 1: WXXM Model Layers**

## 5.2 WXXM data in OWS-8

During OWS-8, the following data sets have been encoded which are relevant for WXXM:

### 5.2.1 Data from CSISS,GMU

- NOAA GOES Inflight Icing product
- NOAA GOES Fog/Low Clouds product

- NOAA GOES Clear Air Turbulence product
- NOAA GOES Volcanic Ash product
- NOAA GOES Wind product
- NOAA GOES Aviation products shown in Google Earth (last updated April 04, 2011):
    - KML example 1: http://geobrain.laits.gmu.edu/ows8/kml/20110405wind/wind20110405.kml
    - KML example 2: http://geobrain.laits.gmu.edu/ows8/kml/wind.kml
- Reference: http://www.star.nesdis.noaa.gov/smcd/opdb/aviation/
- Introduction for WCS for NOAA GOES Wind products: http://geobrain.laits.gmu.edu/ows8/aviationWCS.html

#### 5.2.2 Data from NCAR (contact: Aaron Braeckel)

Two files from the RUC weather model, which are available at http://www.ral.ucar.edu/staff/braeckel/ows8/ruc-model/.

This is generated on a CONUS geographic scale at roughly 20km resolution, where each file represents an entire forecast run. For example, the 1400 file was run at 1400 and generated an analysis of current conditions (can be considered a 0-hour forecast in some ways) as well as forecasts into the future. Therefore, these two files are representing different forecasts for the same actual time.

## 6 WXXM 1.1.1 Architectural Overview

### 6.1 WXXM Audit

The development and improvement of the WXXM 1.1.1 is by further analysis of the architecture and the WXXS exchange schema. The version for analysis in this test bed is WXXM Version 1.1.1.

#### 6.1.1 Overview

The WXXM weather data model, as shown in Figure 2, is designed in three tiers. The base layer is based on weather concepts and supporting packages such as ISO 19100.

This layer has a hierarchical structure of data types, measurements, phenomena, observations and forecasts. The geospatial aspects of WXXM, the physical data model, and the WXXS exchange schema are catered for by use of Geographic Markup Language (GML). Conformance to the GML specification will allow compatibility and use of other standards such as AIXM. WXXM 1.1.1 has been developed using ISO 19136:2007 (GML 3.2.1).

Compliance of WXXM 1.1.1 to ISO 19136:2007 (GML 3.2.1) was approached by two processes, a manual scan and an automated scan. The manual scan involved a review of all schema objects and test conformance to GML. The Galdos® GML SDK performed the automatic scan and has a GML validation feature which checks criteria for compliance. Both scans uncovered no critical compliance issues.

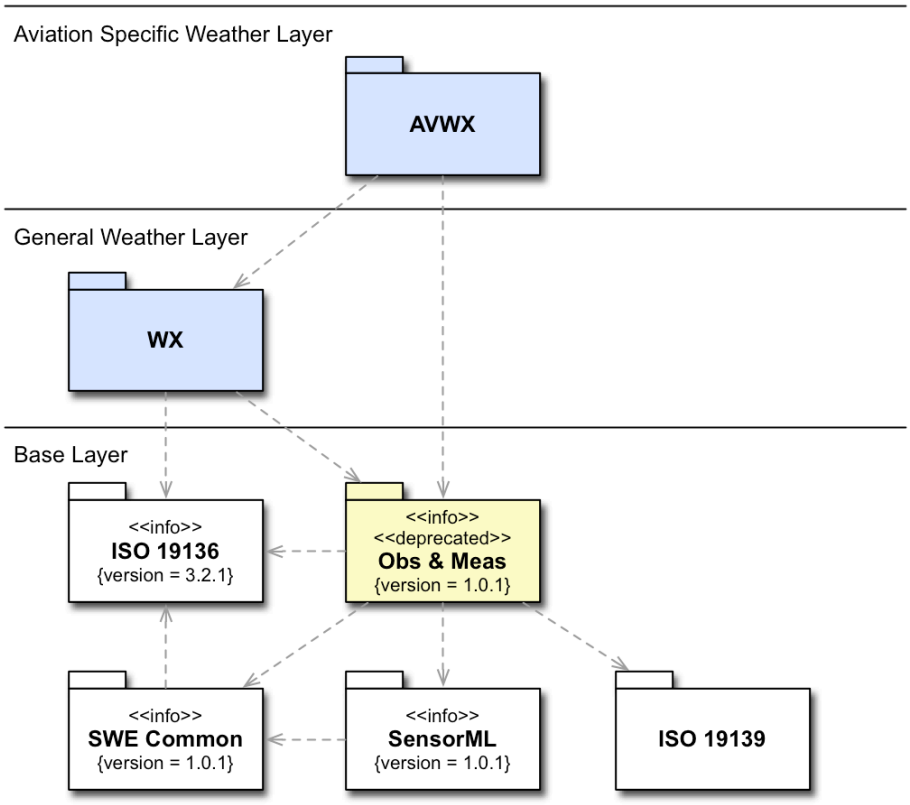


**Figure 2: WXXM Weather Data Model**

The high-level WXXS schema dependencies are illustrated in Figure 2: WXXM Weather Data Model. Only dependencies that span namespace boundaries (i.e., imports) are shown.

 

### 6.1.2 Design Considerations

The ISO 19136:2007 (GML 3.2.1) standard stipulates rules and recommendations regarding the construction of GML application schemas; these constraints are documented in the following clauses:

(a) Clause 7.1: GML model and syntax

(b) Clause 21: Rules for GML application schemas

(c) Annex A.1: Abstract test suite for GML application schemas

### 6.1.3 Deployment

A team at La Trobe University Melbourne has carried out a mapping of Australian weather messages from a textual format into an XML representation to investigate the issues around the development of a collaborative decision support system, particularly the integration of information in the Aeronautical Information Exchange Model (AIXM) and the Weather Exchange Model (WXXM).

The Australian briefing system publishes ICAO Annex 3 weather messages to the Australian industry via many methods. The messages used for the work at La Trobe University were intercepted from the Aeronautical Fixed Telecommunications Network (AFTN). Message types include TAF, METAR, SPECI, TTF and AREA messages.

#### 6.1.3.1 Design Considerations

An Interface Control Document (ICD) published for weather message dissemination in Australia defines the grammar for each message type. An XML schema was designed to correspond to the structure of the message grammars for each type in the ICD document with some inclusions. Message instances based on this schema were transformed by the use of XSLT. TAF messages were used as a test case. Figure 3 shows a snapshot of a WXXM conversion for an Australian TAF message.

- The XSLT sheet contains the mapping structure from text into WXXM
- No data loss tolerated (capture of AFTN header etc.)
- Extensions to WXXM schema allowable

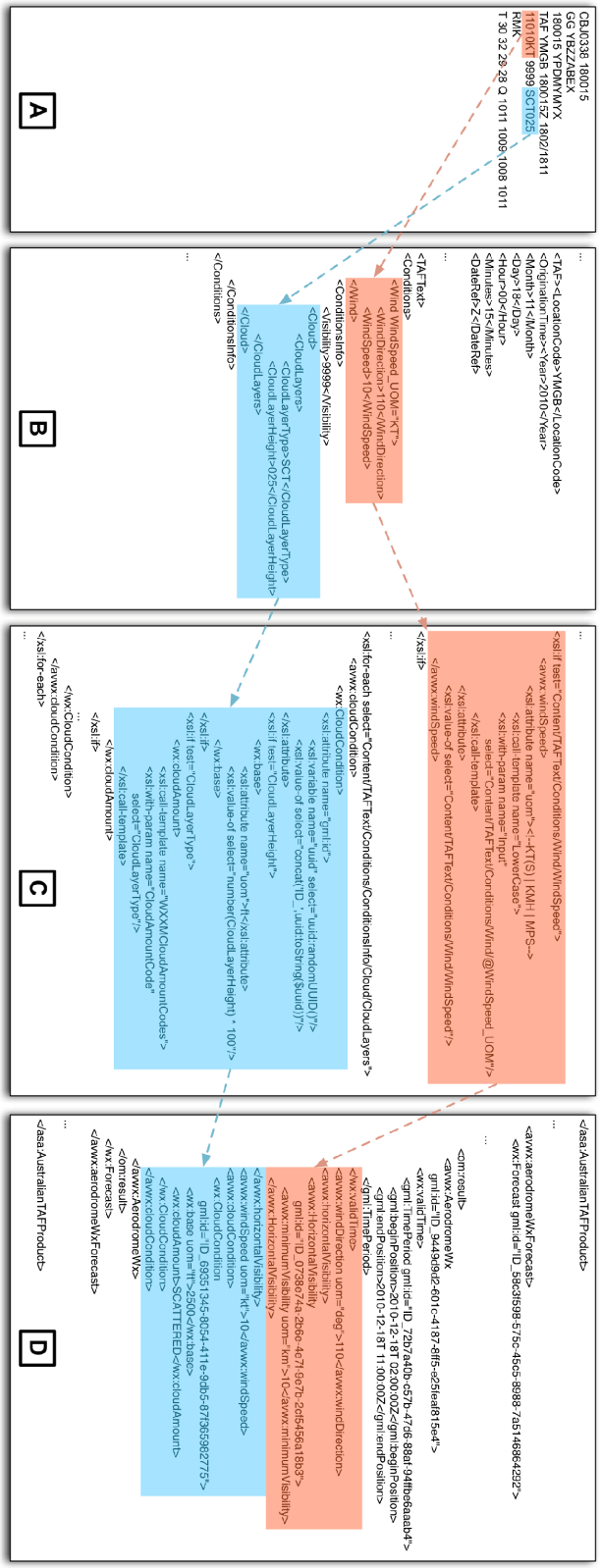

**Figure 3 Mapping Process**

#### 6.1.3.2 Deployment Issues and Recommendations

##### 6.1.3.2.1 Schema Extensions

Schema extension was required to capture the message grammars published for the Australian data.

- Provisional TAF
  In Australia an aerodrome report can be deemed provisional if it is likely to be deficient in accuracy by virtue of its origination.

- Capture Australian textual descriptors describing cloud, weather and wind information

##### 6.1.3.2.2 Non ICAO forecast information

Australia publishes aerodrome forecasts which include forecasts as per ICAO Annex 3. Exploiting of the remark area (**RMK**) of the message, a 12-hour period projection for the temperature and QNH are included.

##### 6.1.3.2.3 Enumeration Conflicts

Enumerations for Australian data are mapped to WXXM with a high level of correspondence. However, some groups of enumerations were in conflict or needed an extension for weather, cloud and turbulence information.

### 6.1.4 Issues and Recommendations

#### 6.1.4.1 Overview

Issues and recommendations are classified under the following headings:

- General Issues
  Issues that may impact more than one domain of the WXXM model.

- Generic Weather Domain Issues
  Issues relevant to the generic weather sub-domain of WXXM.

- Aviation Weather Domain Issues
  Issues relevant to the aviation sub-domain.

#### 6.1.4.2 Specific Rules in Naming GML Schema objects

- Object property rule – no object element (i.e. an element whose type ultimately derives from gml:AbstractGMLType) shall be a child of another object element

 

- UpperCamelCase naming convention for Object element names
- UpperCamelCase naming convention for Complex Type names
- lowerCamelCase naming convention for Property names
- Type name suffix convention 'Type' should be at the end of Complex Type name
- Abstract element naming conventions 'Abstract' should be at the beginning the element name

#### 6.1.4.3 General Issues

- No version information is provided for any schemas.

- Resolution of relative XLink references. The xml:base attribute is not declared in any WXXS schema. A consequence of this is the resolution of relative XLink references, in particular those that are not same-document references. This may be problematic in cases such as when data is retrieved via WFS.

- No object property rule violations were identified by the automated scan.

#### 6.1.4.4 Generic Weather Domain Issues

- Abstract features in wxCoverage.xsd nominally can be instantiated: *wx:AbstractCoverageDomain*, *wx:AbstractCoverage*.

##### 6.1.4.4.1 GML Naming Violations

**Table 1: WX Non Nami*ng Conformance to GML***

| *Sch*em*a* | *Name* | *Amendment* | *Vi*olation |
|---|---|---|---|
| wxPhenomBase.xsd | VerticallyBoundedPhenomenon | AbstractVerticallyBoundedPhenomenon | Abstract Naming |
| wxCoverage.xsd | domainSetPropertyType | DomainSetPropertyType | Complex Type |
| wxGeometry.xsd | geometryPropertyType | GeometryPropertyType | Complex Type |

 

#### 6.1.4.4.2 Spurious Imports and Includes

**Table 2: Redundant Imports and Includes**

| Schema | Import/Include | Issue |
|---|---|---|
| wxForecast.xsd | Import | No dependency on swe.xsd (WXXS does not directly use any SWE elements at all) |
| units.xsd | Import | No dependency on gml.xsd |
| wxMeasures.xsd | Include | units.xsd already included by measures.xsd |
| wxDataTypes.xsd | Include | No dependency on wxPhenomBase.xsd |
| wxCoverage.xsd | Import | No dependency on cv.xsd |
| wxForecast.xsd | Include | No dependency on wxProcess.xsd |
| wxObservation | Include | No dependency on wxProcess.xsd |

#### 6.1.4.4.3 Disjoint Referencing

The wx:AbstractWxFeature may be related to *any* GML feature which may not be the intent. Nothing can substitute for wx:AbstractMeasure (measures.xsd): Speed, Percentage, Temperature, Distance, Angle, Luminance, Pressure all have corresponding subtypes but are not placed in a substitution group.

- WeatherPhenomenonPropertyType (wxPhenomBase) is defined using the GML property type pattern. However, wx:WeatherPhenomenon does not represent a GML object but simple properties that can never be referenced in the usual manner (i.e. using a shorthand pointer[1]).
- WeatherModifierPropertyType (wxDataTypes) is defined using the GML property type pattern. However, wx: WeatherModifier does not represent a GML object but simple properties that can never be referenced in the usual manner.
- AbstractMeasurePropertyType (measures.xsd[2]) is defined using the GML property type pattern. However, wx:AbstractMeasure is not a GML object and can never be referenced in the usual manner.

[1] Alternative XPointer schemes might be used in some contexts.

[2] This also applies to all property types defined in measures.xsd and wxMeasures.xsd.

- wx:Forecast/relatedObservation may refer to *any* GML feature, which may not be desirable.

#### 6.1.4.4.4 Redundancies and General Issues

- Duplicated (anonymous) type definitions in wxForecast.xsdand in other WX schemas. Observations and Measurements 1.0 [3]is deprecated. Consider upgrading to the current v2.0 release (OGC 10-025r1), which uses GML 3.2.1 so an "adapter" is not required.
- A circular dependency exists between the wxBase and wxPhenomBase schemas. While this is not a critical shortcoming, these are best avoided as they can pose complications for profilers.

  With this change, the (include) dependencies among the WX schemas are shown in Figure 4 Weather Domain Schema Dependencies.

[3] om1:Observation has been renamed to om2:OM_Observation

 

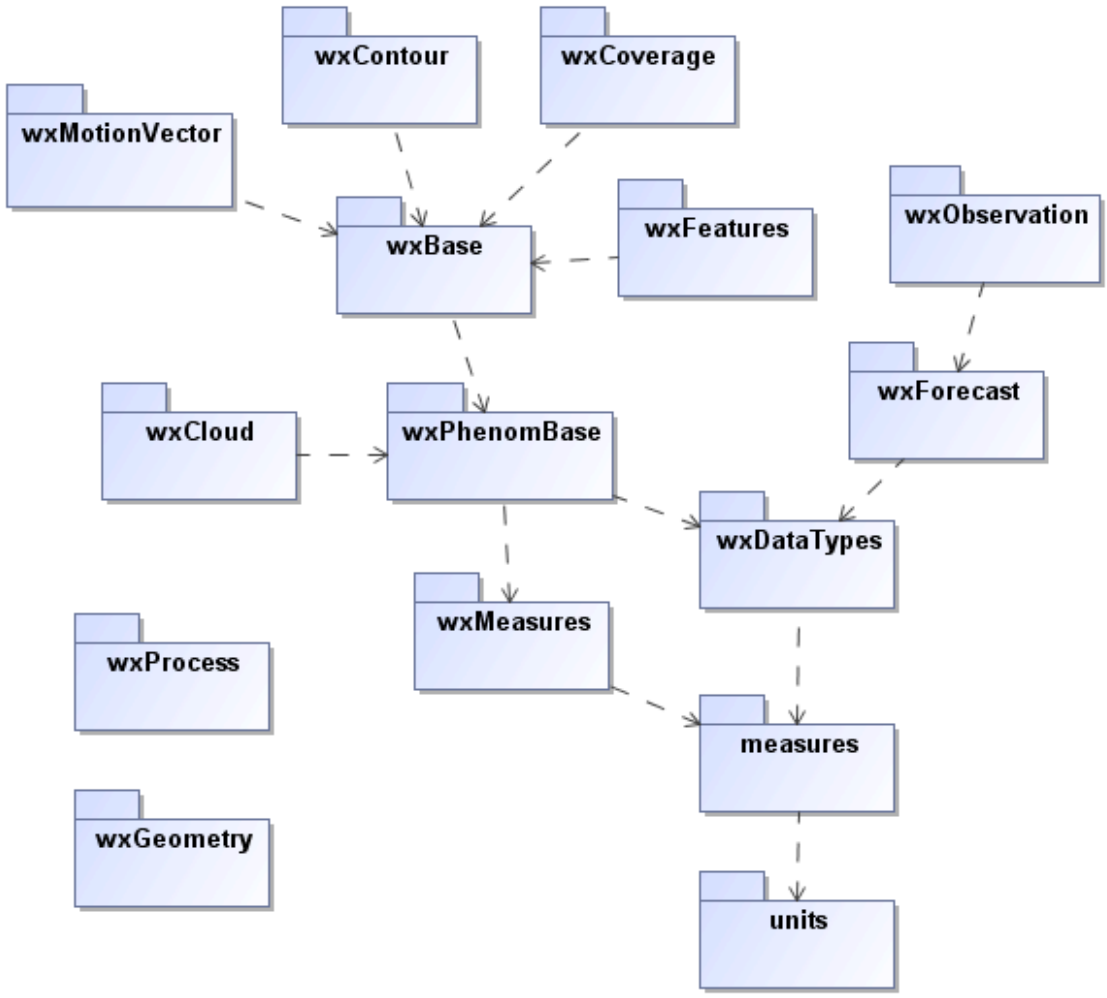


**Figure 4 Weather Domain Schema Dependencies**

#### 6.1.4.5 Aviation Domain Issues

##### 6.1.4.5.1 GML Naming Violations

**Table 3: AVWX Domain GML Naming Non Conformance**

| Schema | Name4 | Amendment5 | Violation |
|---|---|---|---|
| avwxFeatures.xsd | runwayDesignator6 | RunwayDesignator | Object Name |
| icaoAnnex3Base.xsd | TREND | Trend | Feature Type |
| icaoAerialReport.xsd | AIREP | Airep | Feature Type |
| icaoAerialReport.xsd | AMDAR | Amdar | Feature Type |
| icaoAerialReport.xsd | PIREP | Pirep | Feature Type |
| icaoAerialReport.xsd | UrgentPIREP | UrgentPirep | Feature Type |
| icaoAerialReport.xsd | MDCR | Mdcr | Feature Type |
| icaoSurfaceReport.xsd | METAR | Metar | Feature Type |
| icaoSurfaceReport.xsd | METARSpeci | MetarSpeci | Feature Type |
| icaoAreaReport.xsd | AIRMET | Airmet | Feature Type |
| icaoAreaReport.xsd | G-AIRMET | G-Airmet | Feature Type |
| icaoAreaReport.xsd | CCFP | Ccfp | Feature Type |
| icaoAreaReport.xsd | SIGMET | Sigmet | Feature Type |
| icaoAreaReport.xsd | G-SIGMET | G-Sigmet | Feature Type |
| icaoSurfaceForecastReport.xsd | TAF | Taf | Feature Type |

##### 6.1.4.5.2 Spurious Imports and Includes

- Spurious import in avwx.xsd: there is no apparent dependency on swe.xsd in any AVWX schemas.
- Spurious import of gml.xsd in avwxDataTypes.xsd.
- Several spurious include dependencies appear. The actual schema dependencies are shown in the Figure 5 package diagram. Note the circular dependency involving the avwxAreaOfInterestWx, avwxPhenomVolcanic, and avwxFeatures schemas.

---

[4] Expand the acronym if appropriate

[5] Also update all related type definitions accordingly

[6] The Runway/runwayDesignator property has type xsd:string, not RunwayDesignatorPropertyType as one might expect. Should it?

 

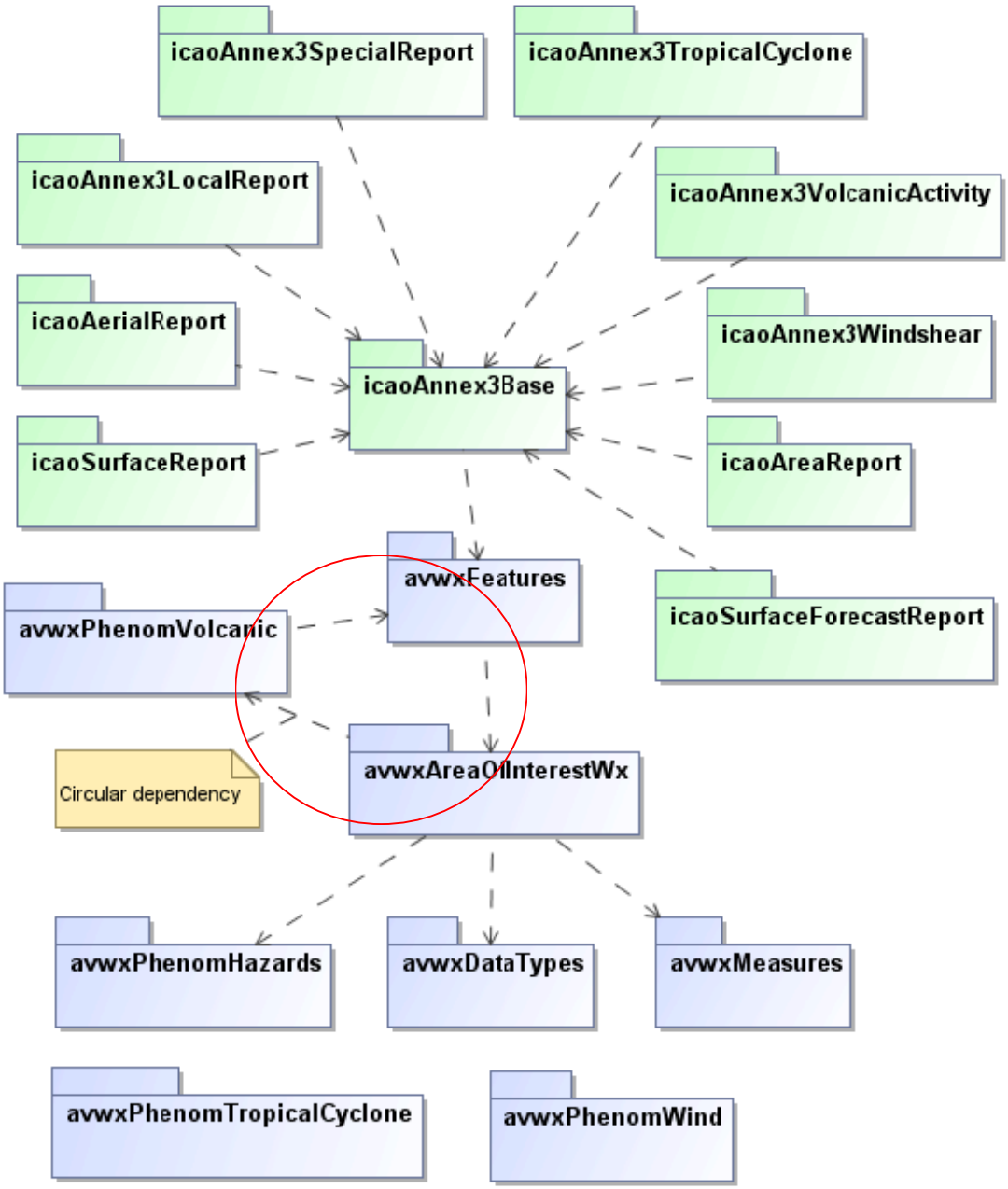


**Figure 5: WXXM Schema Dependencies**

#### 6.1.4.5.3 Disjoint Referencing

- Incorrect reference identified for SpecialAirReport/reportedLocation (in icaoAnnex3SpecialReport.xsd). This property is a link reference of type gml:ReferenceType. However, the expected value (gml:targetElement) is gml:posList, which is not a GML object.

.

- Unfounded GML property types defined in avwxMeasures. The references are neither GML objects nor possess schema-determined identifiers.
- Profligate use of anonymous property types in icaoAnnex3VolcanicActivity.xsd, icaoAnnex3Windshear.xsd, icaoAreaReport.xsd, icaoSurfaceForecastReport.xsd .
- VolcanicActivity/volcano property (in avwxPhenomVolcanic.xsd) has an anonymous type definition that resembles avwx:VolcanoPropertyType.
- VolcanicActivityReport (icaoAnnex3VolcanicActivity.xsd) makes no use of the schema components in avwxPhenomVolcanic.xsd.

#### 6.1.4.5.4 Redundancies and General Issues

- The element declaration “isDefinedFor” (in avwxAreaOfinterestWx.xsd) has an anonymous type definition that is not consistent with the Schematron assertions.
- Redundant name property in avwx:TropicalCyclone[7]. The type inherits gml:name (0..*) from gml:AbstractGMLType.
- TropicalCycloneAdvisory/number property (icaoAnnex3TropicalCyclone) allows negative integers.
- Redundant description property on VolcanicActivityType in avwxPhenomVolcanic.xsd. The type inherits gml:description from gml:AbstractGMLType.

### 6.1.5 WXXM Coverage Representation

#### 6.1.5.1 Introduction

WXXM extensively uses coverages. These are defined in the WXXM WX_Coverage package as one possible implementation of the ISO 19123 coverage model.

[7] This also applies to the following types defined in avwxFeatures.xsd: MeteorologicalWatchOfficeType, AirspaceType, AerodromeType, SeaType, VolcanoType, VolcanicAshAdvisoryCentreType, and TropicalCycloneAdvisoryCentreType.

 

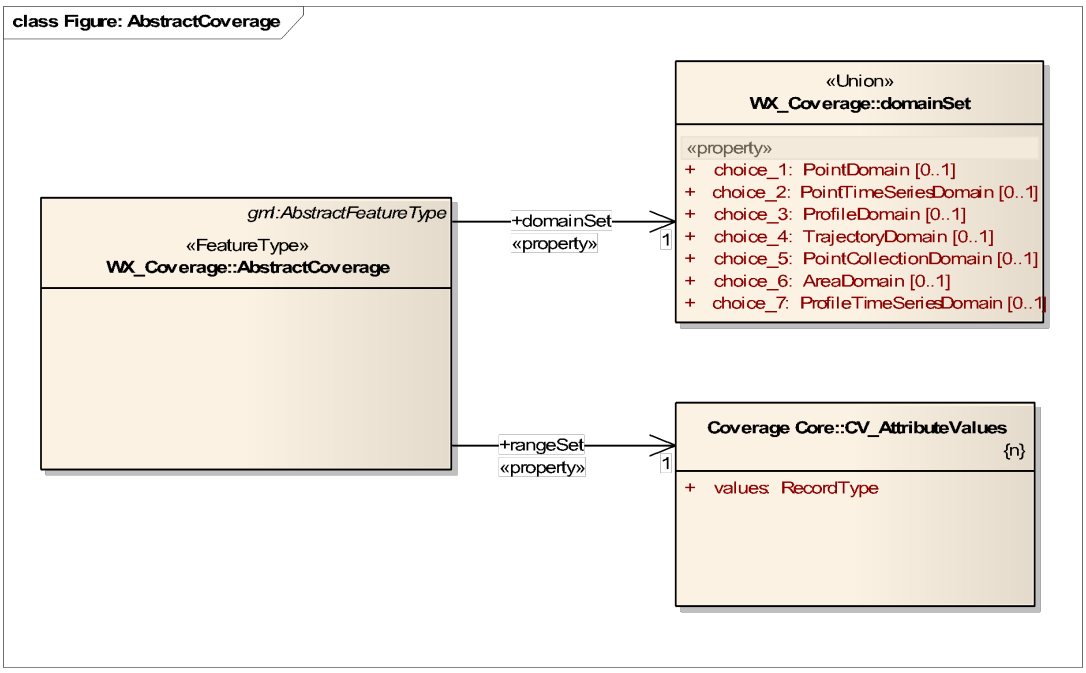


**Figure 6: The WXXM Coverage structure**

Figure 6 shows the concrete (instantiable) coverage types defined in WXXM.

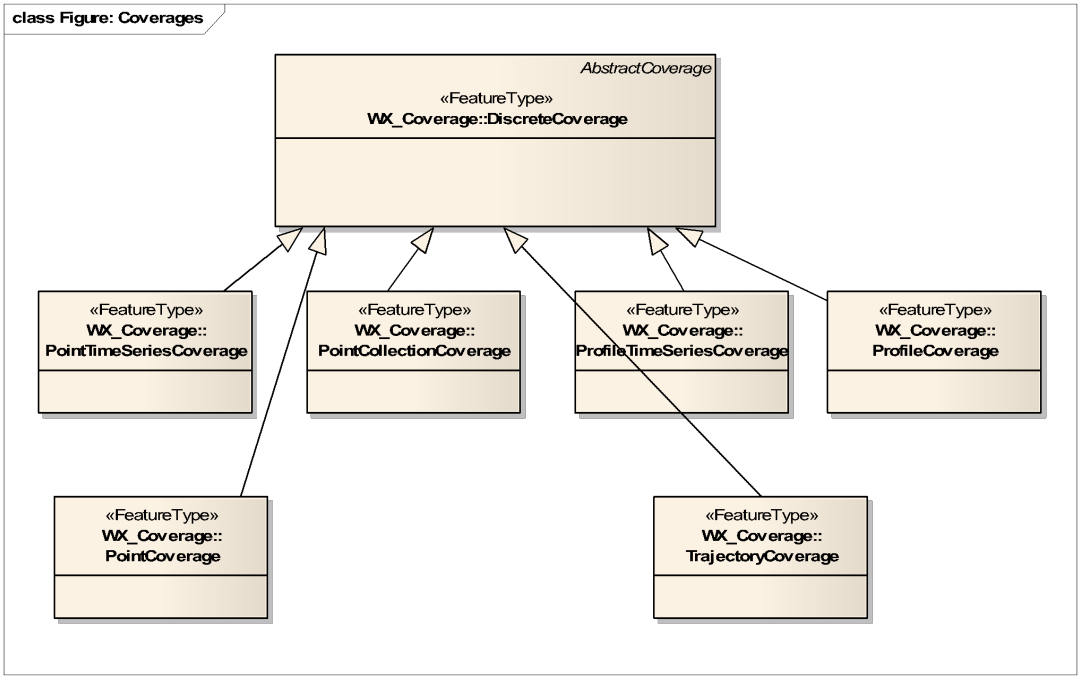


**Figure 7: The WXXM coverage hierarchy**

With the WXXM coverage hierarchy shown in Figure 7, it appears that it doesn't align closely with the ISO 19123 definitions. For example, the WXXM PointCoverage corresponds to the ISO 9123 CV_DiscretePointCoverage; WXXM ProfileTimeSeries does not have a correspondence in ISO 19123; etc.

The Northrop Grumman Technical Paper suggests that "the main issue in dealing with the above types of data has been that the ISO coverage model specifically restricts coverages to 4D (1-3 spatial dimensions and 0-1 time dimension)". This report is specifically focused in scope on dealing with the two data types (i.e. Forecast model runs and Radar data) and encoding them into an ISO 19123:2005 compliant coverage. The position postulated by the weather community, referred to in Annex B: OWS-8 section 4.5.2.3.2 is that if additional dimensions are required in order to record this information, the resulting artifact would be non-compliant with the ISO 19123:2005 coverage model.

Here again the question is that of domain versus range. The ISO 19123:2005 specification limits the domain (collection of direct positions) to the four spatial-temporal dimensions as previously noted.

On a side note, there seem to be some copy-and-paste errors resulting from copying and modifying GML schema code. For example, DiscreteCoverage (wxCoverage.xsd) is documented as "*This element serves as the head of a substitution group which may contain any discrete coverage whose type is derived from gml:DiscreteCoverageType*" whereas this element certainly is not meant to be in the gml namespace.

#### 6.1.5.2 Comparison with OGC coverage model, GMLCOV

Both WXXM and GMLCOV share the notion of a coverage consisting of domain and range. Further, the approach of sub-typing an abstract coverage into instantiatable concrete coverage types is common to both. The coverage types provided by WXXM and GMLCOV are compared in Table 4. Note that this does not mean schema compatibility, rather this inspects informational contents.

**Table 4: WXXM vs. GMLCOV coverage types**

| WXXM | GMLCOV |
|---|---|
| PointTimeSeriesCoverage | Special case of MultiCurveCoverage |
| PointCollectionCoverage | Special case of MultiPointCoverage |
| ProfileTimeSeriesCoverage | *Some grid coverage type?* |
| ProfileCoverage | *Some grid coverage type?* |

 

| | |
|---|---|
| PointCoverage | MultiPointCoverage |
| TrajectoryCoverage | MultiCurveCoverage |
| *No correspondence* | RectifiedGridCoverage |
| *No correspondence* | ReferenceableGridCoverage |
| *No correspondence* | GridCoverage (Deprecated) |

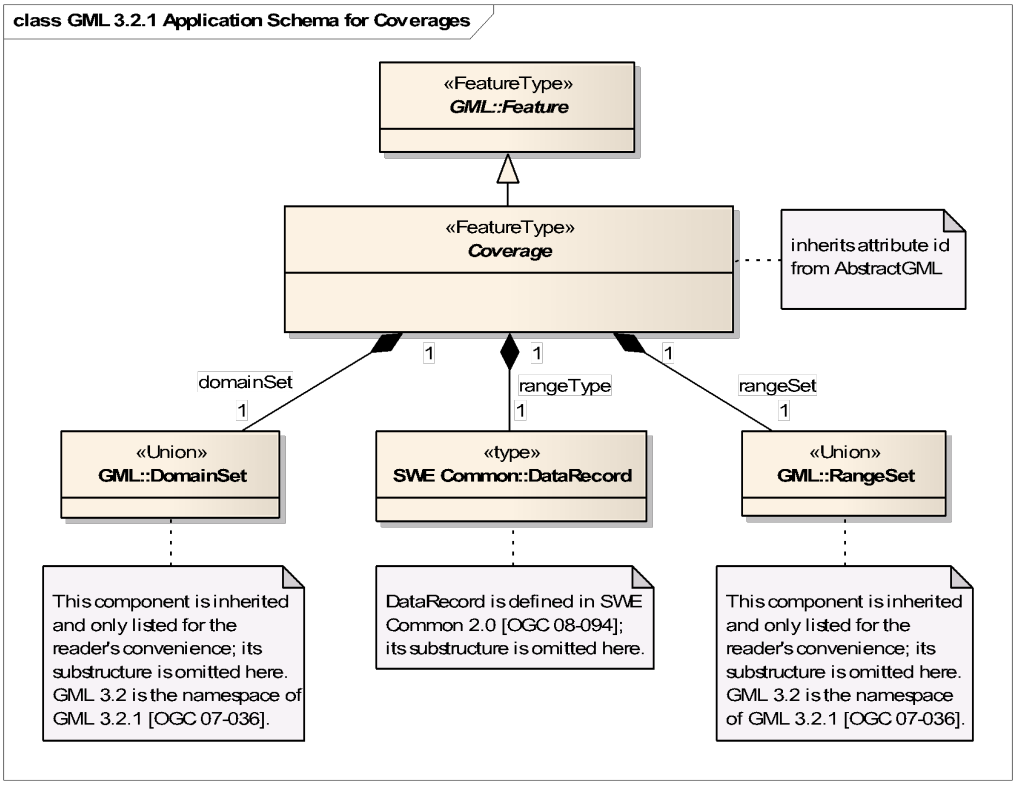


**Figure 8: The GMLCOV Coverage structure**

The major differences between the two models can be summarized as:

- WXXM coverages do not have a range type definition like GMLCOV. This element has been added to provide comprehensive information about the range ("pixel") data structures. Hence, the WXXM coverage is informationally incomplete.

- The domain model is very specific; it will not translate nor embed into a general coverage model. It is stated that "This element is based on the GML 3.2 version, with the exception that it has been extended to allow for a combined

 

spatial/temporal domain (as per ISO 19123)". While GML already considers time, the new element actually is not compatible when disregarding the extra item added.

- Another example for incompatibility is the special treatment of z coordinates codified into element names, rather than explicit semantics:

```
<element name="timePositionList" type="wx:TimePositionListType"/>
    <element name="locationZ" type="gml:DirectPositionListType"/>
    <element name="location" type="gml:DirectPositionType"/>
```

  This will lead to problems in the context of n-D coverages and CRS handling.

- The range set is modeled as a SWE Common DataRecord. This actually allows more than the set and sequence concepts of coverages, but also, for example, structs and arrays, as well as arbitrary nesting. As such, it transcends the idea of a coverage and is incompatible with GML and GMLCOV.

- In summary, the WXXM model is different from the OGC definition of coverages, as laid down in [OGC 10-146r1] and in the sequel abbreviated as GMLCOV, to the extent that it indeed is incompatible. In other words, implementations adhering to said OGC coverage definition will not be able to produce or consume WXXM coverages, and vice versa.

- On a side note, it is a common misconception that relying on ISO 19123 yields interoperable coverage definitions. ISO 19123, by its purpose and definition, establishes an abstract coverage model on which many diverging implementations can be based. To achieve concrete interoperability, one such concrete implementation model has to be chosen.

#### 6.1.5.3 Recommendation

It is recommended to base the WXXM coverage concept on the common OGC definition of coverages to achieve interoperability at the document and service level. Technically, this means replacing the coverage root WX_AbstractCoverage by GMLCOV::Abstract-Coverage. Those coverage types not yet present in the (generic) OGC coverage model might be introduced, in collaboration with the WCS.SWG. In principle, there are two options:

- Define an Application Profile specification (as has been done in the Earth Observation domain with EO-WCS).

- Write a Change Request to add further coverage types to GMLCOV.

The optimal approach, which may involve a combination of both, needs to be elaborated with the Web Coverage Service (WCS) Standards Working Group (WCS.SWG) and the

 

various stakeholder groups. According to EO-WCS experience, XML definitions can be conveniently crafted through a combination of XML Schema and Schematron.

Generally, it is strongly recommended to liaise with the WCS.SWG for coverage modeling, as the WCS.SWG is the central maintenance point for coverages in OGC. This ensures harmonization and interoperability across communities.

This should not constitute a devaluation of the work accomplished. On the contrary, the contributions made by WXXM are of great value in the further development of the concept of coverages for manifold practical use cases.

**Consequences if not done:**

- WXXM uses the term “coverage” without compatibility to OGC coverage concept; this can lead to confusion in the target communities and, hence, is adverse to OGC’s mission.
- WXXM coverages cannot be used interchangeably with other coverages in OGC at the instance level.
- WXXM coverages cannot be served through OGC conformant WCS services, plus further services like WCPS, EO-WCS, forthcoming WPS coverage processing application profile, and others in future.
- In particular, WXXM cannot make use of the coverage encoding format extensions currently under development within OGC (ie, in OWS-8), as these are fitted into GMLCOV.

## 6.2 Unit of Measure (UOM)

### 6.2.1 Overview

We identify three categories of issues in relation to distributed Unit of Measure (UOM) namely the **encoding**, **deployment**, and **interoperability** of UOM. We define these groupings as part of the weather’s group thoughts, discussions and incremental explorations on the issues surrounding distributed UOM in general. Each of these categories will be discussed in the following sections.

#### 6.2.1.1 Encoding Allignment

The first issue is the encoding of the distributed UOM. This has been addressed in several earlier references. In principle, basic UOM is coded as an attribute of element/s, which referenced a global definition of the UOM. The referenced UOM can be either a pre-defined UOM (also known as base UOM or conventional UOM) and therefore it is already fully defined in the global dictionary, or is a new UOM.

 

The pre-defined UOM may be:

- based on the International Systems of Units (SI), or other conventional /well-known systems for general geospatial domains.
- based on the unit of measures defined in the underlying model of the application, e.g. Weather Exchange Model unit of measures for weather related applications.

**New UOMs** that are not specified in the pre-defined dictionary will have to be '**derived'** from the existing pre-defined unit. The new UOM will be represented in a **UOM repository** together with the conversion/derivation method.

The standard representation of the UOM encoding remains an issue to be convened. The following section is a summary of the contributions of:

https://portal.opengeospatial.org/twiki/bin/view/OWS8/AviationWeatherWxxm#Harmonization
http://metadata.ces.mil/mdr/ns/GSIP/uom
http://metadata.ces.mil/mdr/ns/GSIP/uom/doc
Luciad (Jeroen Dries)
Eurocontrol (Eduard Porosnicu)
MITRE Corporation (Paul A. Birkel)
Jacobs University (Peter Baumann)
NCAR (Aaron Braeckel)

Even though AIXM and WXXM are both based on GML all three use a different measure system. This has already been discussed with the WXXM group in 2010 and in OWS-7. It is widely agreed on that a unification of the measure systems would be preferable. Issues holding back the process are backward compatibility with AIXM 4.5 and uncertainty surrounding the ISO standard forming the basis for UCUM.

The following solutions were proposed:

- The use of (OASIS) code lists for the UOMs and schematron.
- An own namespace for gml:MeasureType.
- The use of UCUM wherever possible.
- Linkage between measures and code list.
- The use standard GML Dictionaries.

 

- To create a new complex type that is a <choice> between a gml:MeasureType and a metadata-type (to cater for values such as UNL or CEILING (in AIXM).

- Allow for authoritative and common unit and measure definitions from multiple authorities (UCUM, US National Weather Service, ICAO, Eurocontrol) plus their extension.

- Allow for distributed, formal definitions (e.g., GML unit dictionaries) that are preferably resolvable (i.e., http://ucum.org/uom/meter).

- Allow for non-numeric and mixed numeric and non-numeric types.

- Allow for "local" (new) definitions. This could perhaps be handled through local authorities.

#### 6.2.1.2 Deployment

The second challenge is the deployment of the distributed UOM. The basic underlying deployment approach is that UOMs are encoded in a UOM repository, registered, and deployed as simple URL or web services, which are accessible by end-users. While this might be suitable for pre-defined UOMs, which have clear authoritative ownership, there is a need for a clearer guideline for the deployment of new UOMs that considers:

- the issue of management, validation, and registration processes of new UOM repositories (created and referenced by different research groups all around the world). There might be a need for a ‘registration' method for a new UOM and a central registry, which hosts a global index of the location/URI of these new UOMs.

- the issue of embedding/exchanging a new UOM Dictionary together with the existing WXXM message that utilizes the new UOM versus referencing a ‘registered’ new UOM repository each time the new UOM is used in a WXXM message.

#### 6.2.1.3 Interoperability

The third challenge is the **interoperability** of the distributed UOMs.

- For example, a user may submit a request/query which requires access to a number of UOM repositories. The idea is to have a mechanism for conversion from one UOM repository to another, depending on the context of the user (e.g. geographical location). This requires another set of services on top of the repositories that can convert or translate from one repository to another.

- In another example, a user may have a ‘preferred’ UOM profile, which requires him/her to view UOM in the preference that has been specified, regardless of the

 

source of the original UOM data provider. This will require a mechanism for conversion from multiple dictionaries to a uniform dictionary.

### 6.2.2 Architectural Design Options for Distributed UOM

In all cases, it is necessary to maintain a central repository (or a number of mirrored repositories) that contain the so-called base UOMs and conventional UOMs. Base UOMs and conventional UOMs are generally agreed on and widely used (e.g. SI-based UOM).

Every other, new UOM must be derived from a base UOM / multiple base UOMs. This derivation needs to be encapsulated into a remotely accessible method, such as a URL or web service. But the question arises as to where to store these new UOMs and their associated/relevant services.

#### 6.2.2.1 Centralized Architecture

A new UOM is registered at one of the global repositories (which then is mirrored). This needs to be a defined process to ensure correctness.
The shareholder of the new UOM also needs to deliver web services to transform their new UOM into base UOM(s). See Figure 9 for the architecture, Figure 10 for the initialization sequence and Figure 11 for the data exchange sequence.

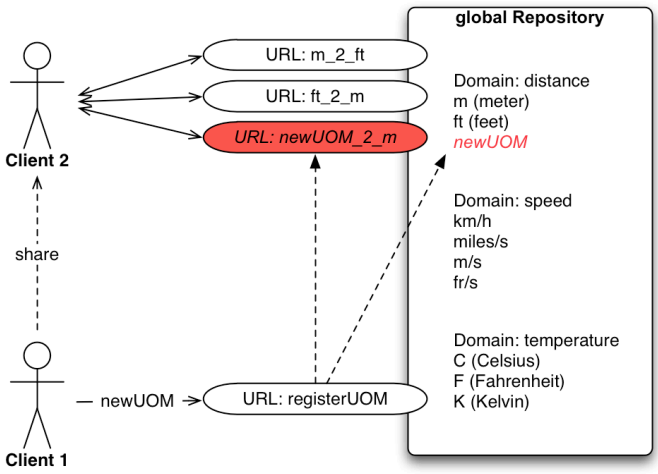


**Figure 9: Centralized Architecture**

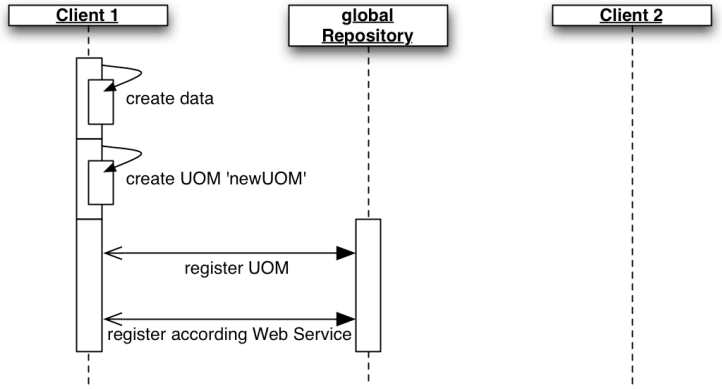


**Figure 10: Centralized Initialization**

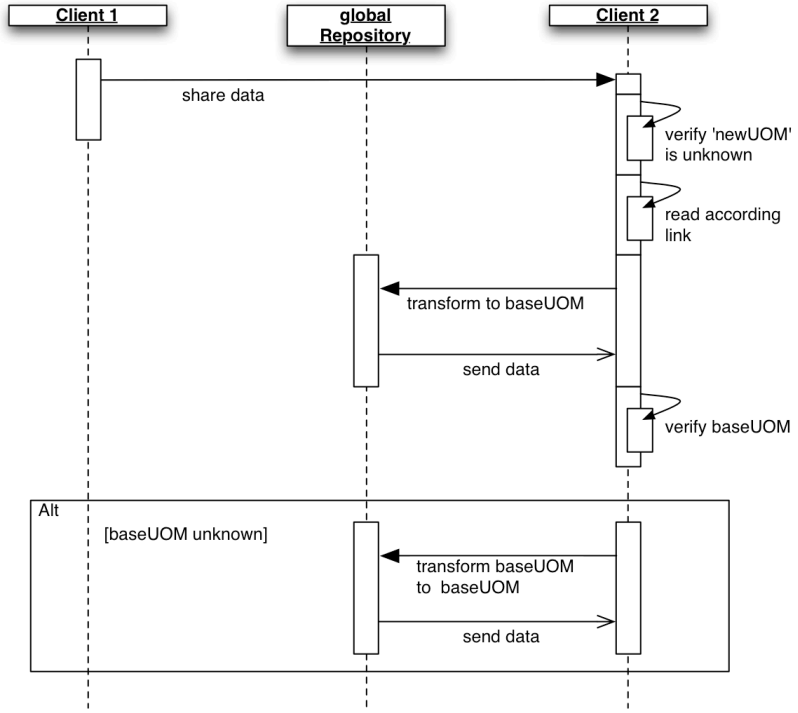


**Figure 11: Centralized Sequence**

#### 6.2.2.2 De-Centralized Architecture

Participants can create their own repository. They "register" the new UOM and the associated services in the local repository.

If data is shared with another client, that client needs to access the participant's local repository for the conversion method. If the new UOM is derived from base UOM(s)

 

unknown to the receiver, it might become necessary to also access the central, global repository for transformation from base UOM to base UOM. See Figure 12 for the architecture, Figure 13 for the initialization sequence and Figure 14 for the data exchange sequence.

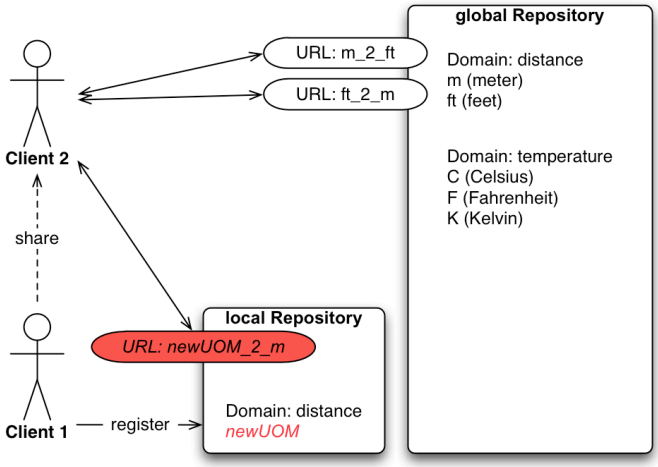


**Figure 12: De-Centralized Architecture**

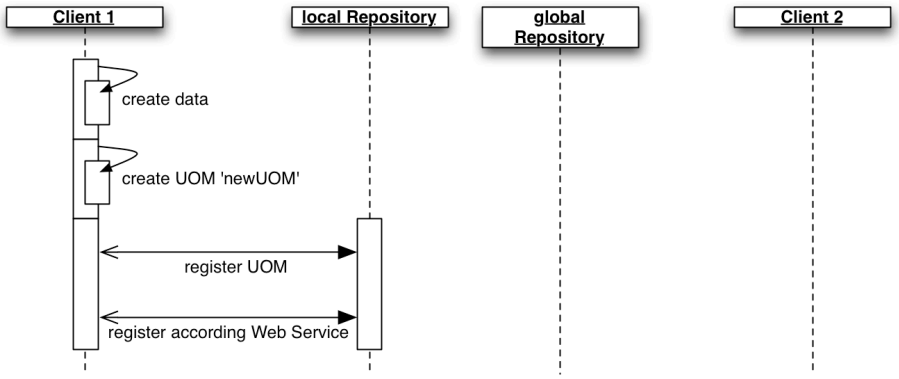


**Figure 13: De-Centralized Initialization**

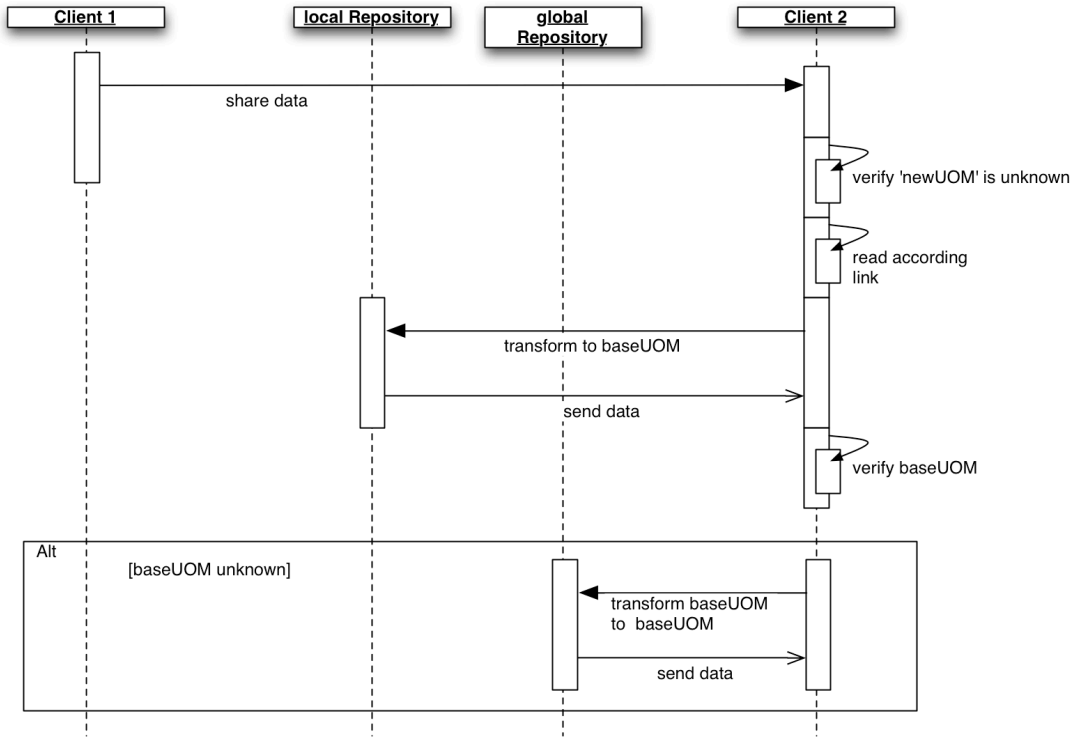


**Figure 14: De-Centralized Sequence**

### 6.2.2.3 Local Copy Architecture

This is a third option which minimizes network traffic. Instead of accessing a web service, the necessary translation functionality is downloaded into a local repository. Whenever the receiver needs to transform this particular UOM in the future, they can access it locally. See Figure 15 for the architecture, Figure 16 for the initialization sequence and Figure 17 for the data exchange sequence.

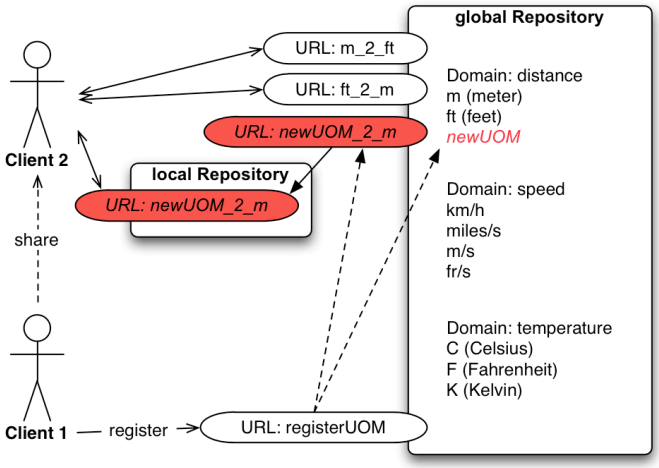


**Figure 15: Local Copy Architecture**

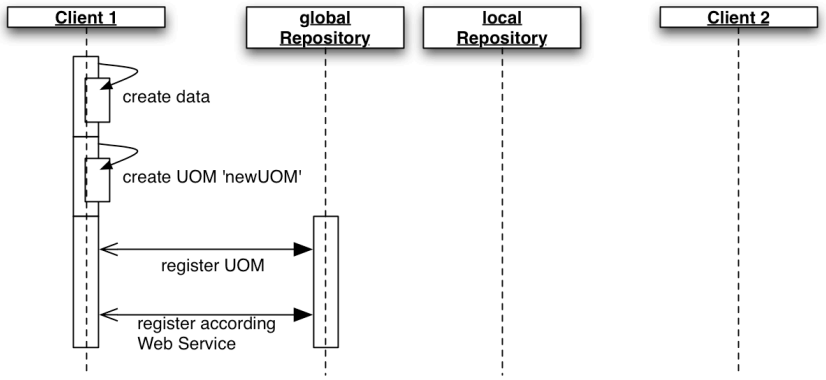


**Figure 16: Local Copy Initialization**

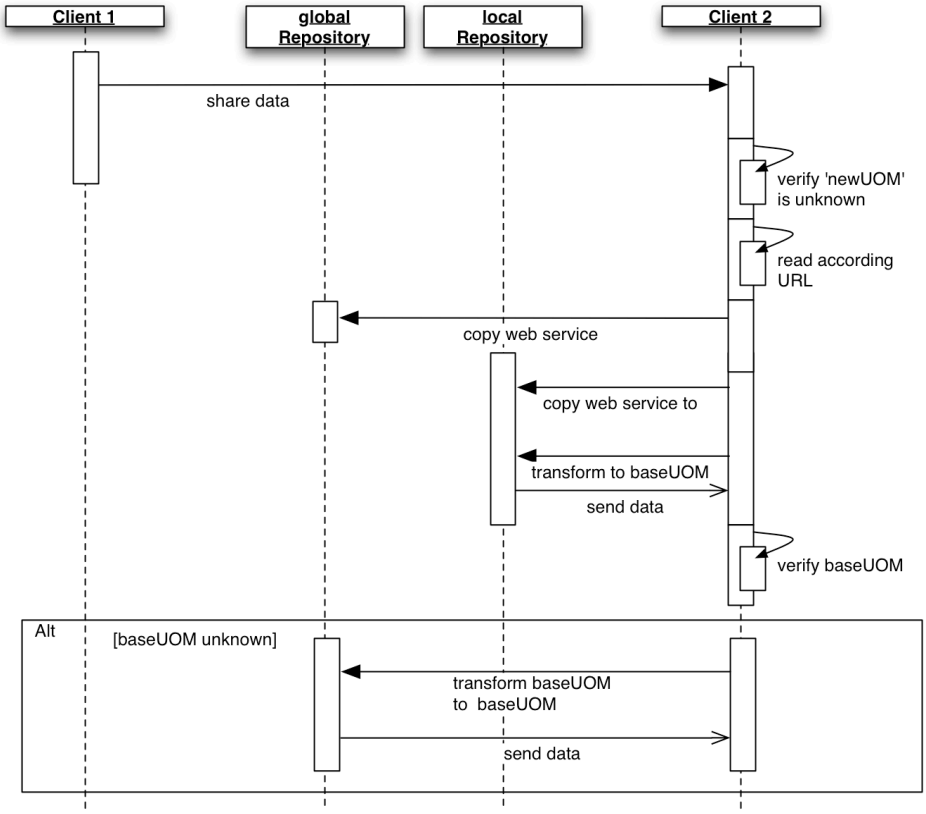


**Figure 17: Local Copy Sequence**

#### 6.2.2.4 Local Transformation Architecture

In this design base, UOMs and conventional UOMs can be thought of as a standard. All transferred data must be encoded in those UOMs. This means the creator of a new UOM has to transform their data before sending it to a broader audience. This step is unnecessary if the sender can be sure that the receiver understands his new UOM (e.g. two research groups sharing information). See Figure 18 for the architecture, Figure 19 for the initialization sequence and Figure 20 for the data exchange sequence.

This approach renders it unnecessary to register and maintain new UOMs in a repository outside the creator's domain. It significantly reduces network traffic and allows for offline applications.

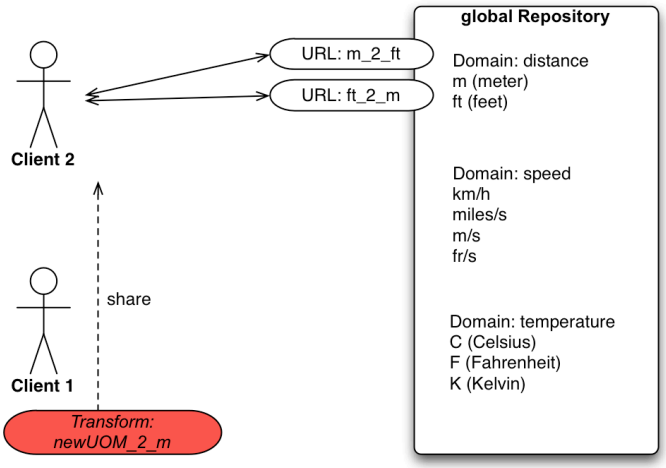


**Figure 18: Local Transformation Architecture**

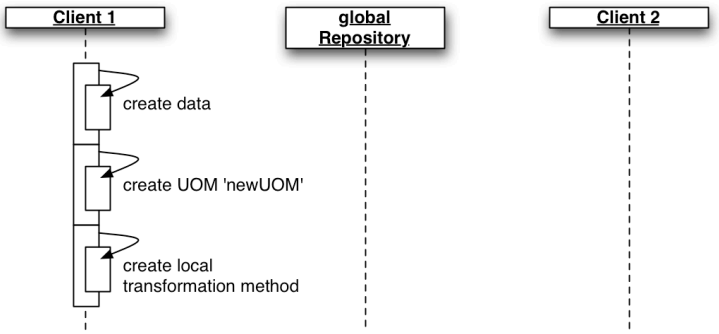


**Figure 19: Local Transformation Initialization**

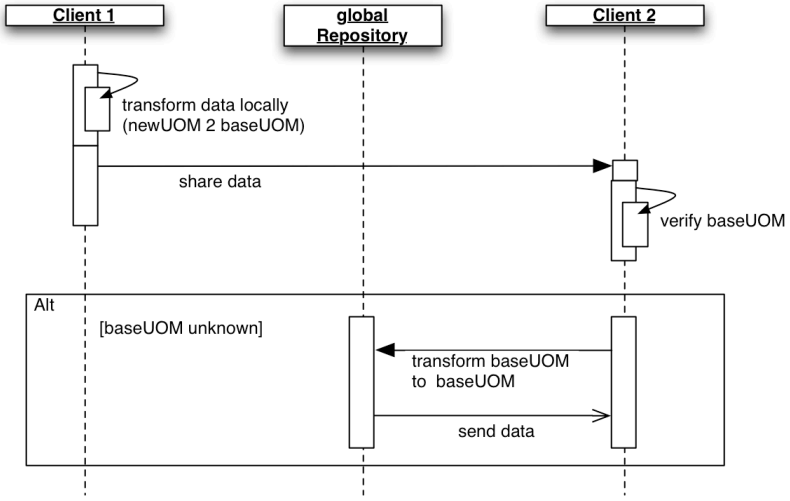


**Figure 20: Local Transformation Sequence**

### 6.2.3 Deployment

#### 6.2.3.1 Separation of semantic and physical layers

The previous sections focused on the physical architectural design that will facilitate the conversion process in a distributed UOM rather than the representation of the distributed UOM itself (i.e. separation of the semantic and logical layers of the conversion process). It has been suggested to establish a 'semantic' or definition layer - separate from the physical deployment layer - which will address the re-use, interoperability and governance issues.

Our exploration takes one form, i.e. the GML dictionary as the encoding representation, simply because we can create scenarios/data sets based on our existing WXXM and AIXM data and explore the possibility of running experiments with the UOM in the data instance itself by following the given UOM GML schema. Other approaches such as UCUM are available as well. Having a generic definition in the semantic layer and then letting local sites deal with their own libraries/services will fit into the "local copy" architecture based on the Design Options as described in section 6.22.

It is necessary to cater for re-use and the potential to reduce governance and management overhead, as well semantic interoperability.

Figure 21 explores the architecture resulting from 'moving up' the logic/conversion into the definition (i.e. semantic) layer, separate from the services/physical layer. This will facilitate the 'local copy' approach, which is created through the forward cache technique of the centralized definition repository.

 

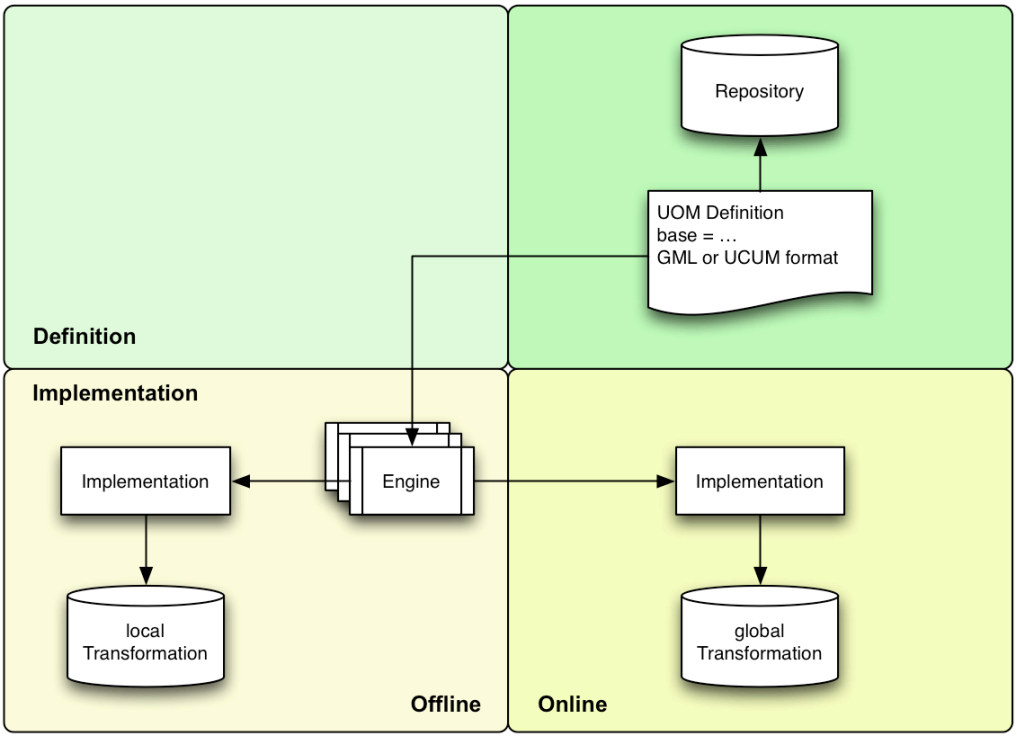


**Figure 21: Greater picture architecture**

#### 6.2.3.2 Data & Schema

It is necessary to find the transformation corresponding to the currently evaluated value/UOM. Hence, every XML element which has a UOM attribute must either include its well-known (i.e. standard) name (e.g. m or ft) or the corresponding link to its transformation web service. Alternatively, one could think of a general link to the repository as an attribute of the root node. See the code listing below and Figure 22 for the schema implementation.

```
<asa:AustralianTAFProduct
    xsi:schemaLocation="http://www.airservicesaustralia.com.au
    C:/wxxm/AustralianProductsSchemas/AustralianTAFproduct.xsd"
    gml:id="ID_46c691d5-aa88-4ca6-9879-1e507f065b32"
    xmlns:uuid="xalan://java.util.UUID" xmlns:gco="http://www.isotc211.org/2005/gco"
    xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:wx="http://www.eurocontrol.int/wx/1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:om="http://www.opengis.net/om/1.0/gml32" xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:avwx="http://www.eurocontrol.int/avwx/1.1" xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:asa="http://www.airservicesaustralia.com.au"
    xlink:href="urlToGeneralWebService">
  <avwx:rawText>
      TAF YMGB 180015Z 1802/1811
      11010KT 9999 SCT025
      RMK
      T 30 32 29 28 Q 1011 1009 1008 1011
</avwx:rawText>
  <avwx:aerodromeWxForecast>
    <wx:Forecast gml:id="ID_58c3f598-575c-45c5-8988-7a5146864292">
```

```
      <om:samplingTime>
        <gml:TimePeriod gml:id="ID_f3bdffee-a2c6-4d05-8097-e32943c1a5f8">
          <gml:beginPosition>2010-12-18T 02:00:00Z</gml:beginPosition>
          <gml:endPosition>2010-12-18T 11:00:00Z</gml:endPosition>
        </gml:TimePeriod>
      </om:samplingTime>
      <om:result>
        <avwx:AerodromeWx gml:id="ID_9449d9d2-601c-4187-8ff5-e25feaf815e4">
          <avwx:windDirection uom="deg">110</avwx:windDirection>
          <avwx:horizontalVisibility>
            <avwx:HorizontalVisibility gml:id="ID_0738e74a-2b6e-4e7f-9e7b-2cf5456a18b3">
              <avwx:minimumVisibility uom="urlToSpecificWebService">10</avwx:minimumVisibility>
            </avwx:HorizontalVisibility>
          </avwx:horizontalVisibility>
          <avwx:windSpeed uom="urlToSpecificWebService">10</avwx:windSpeed>
          <avwx:cloudCondition>
            <wx:CloudCondition gml:id="ID_69351345-8054-411e-9db5-87f365982775">
              <wx:base uom="urlToSpecificWebService">2500</wx:base>
              <wx:cloudAmount>SCATTERED</wx:cloudAmount>
            </wx:CloudCondition>
          </avwx:cloudCondition>
        </avwx:AerodromeWx>
      </om:result>
    </wx:Forecast>
  </avwx:aerodromeWxForecast>
</asa:AustralianTAFProduct>
```

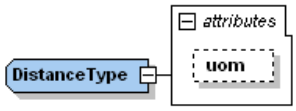


**Figure 22: UOM in the Schema**

#### 6.2.3.3 Transformation

For performance reasons, it is necessary to be able to bypass a remote process and transform data locally. This needs to be decided by the interpreting application. An example of this process can be seen in Figure 23. Section 6.2.3.4 Profiles outlines a possible solution to determine whether a transformation is necessary, and whether it should be performed locally or remotely.

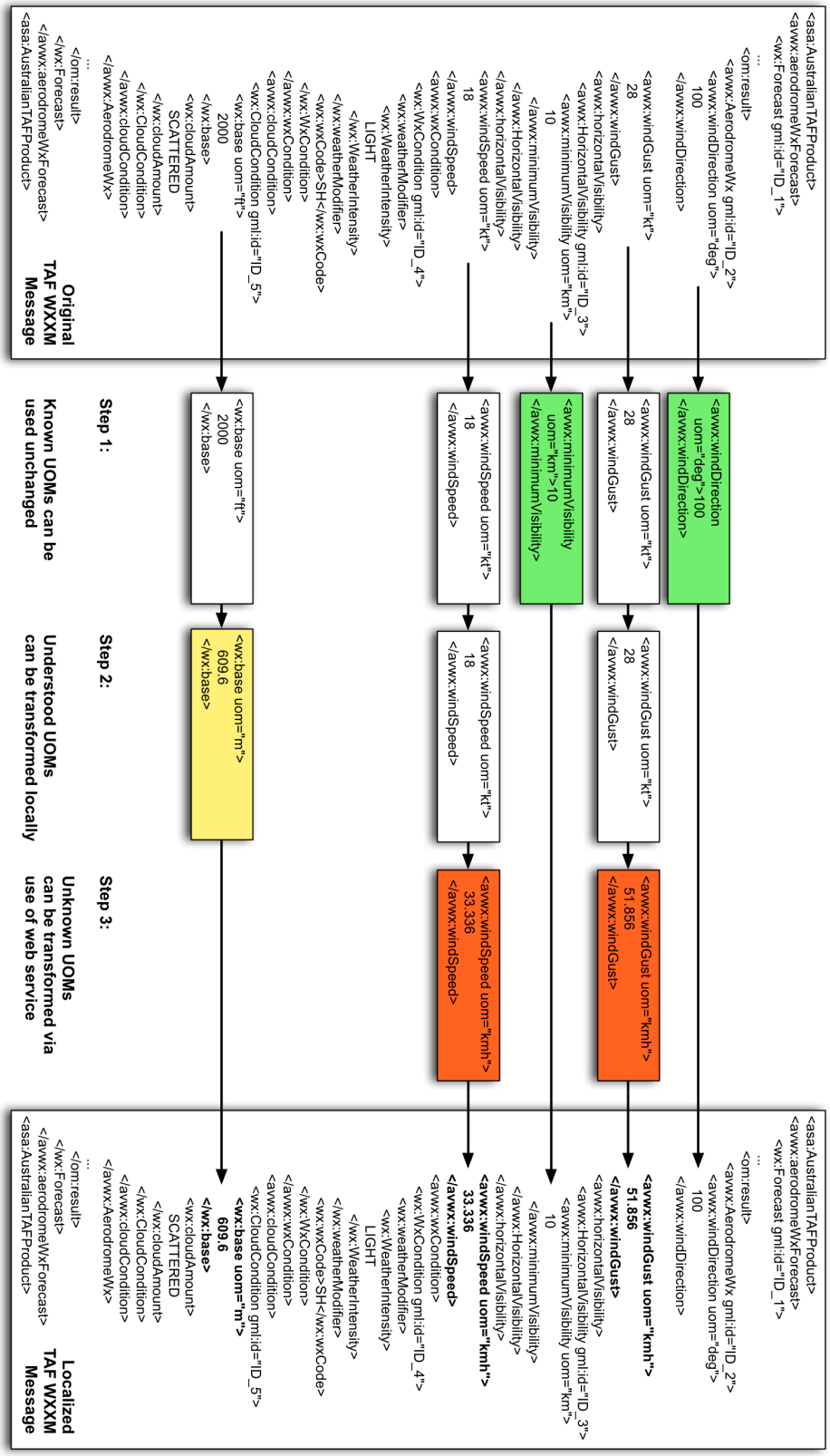

**Figure 23: UOM Transformation step by step**

#### 6.2.3.4 Profiles

UOM Profiles could play a vital role in the area of distributed UOM and would allow the alignment of sender and receiver, which minimizes the number of necessary transformations. This can minimize calculation errors as well. Figure 24 visualizes this approach in the Local Transformation environment.

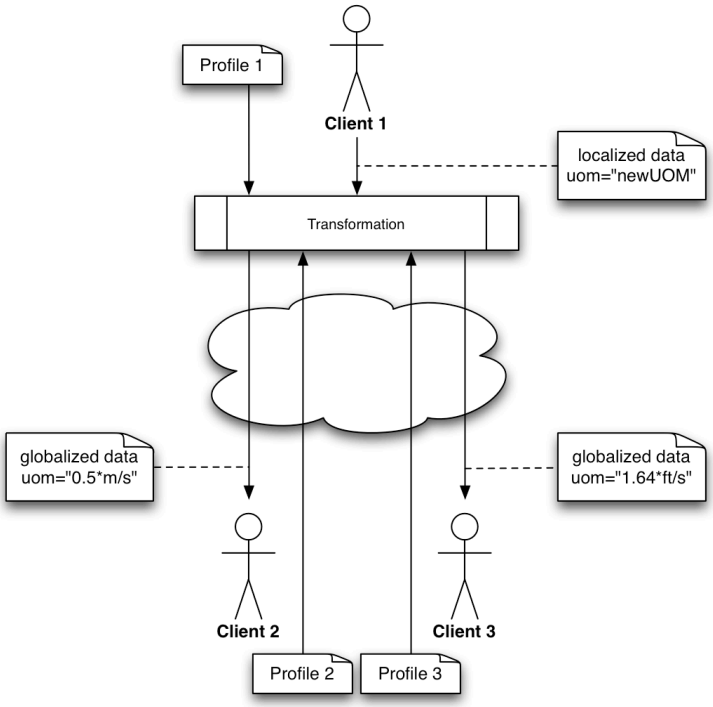


**Figure 24: UOM Profiles**

#### 6.2.3.5 Implementation

The process implementation shown in Figure 25 caters for the challenges highlighted in the UOM section and applies the architecture shown in 6.2.3.1 Separation of semantic and physical layers. The main purpose is to separate the definition of new UOMs from its actual implementation. An application, here a Java Script web application facilitates the deployment of the process.

 

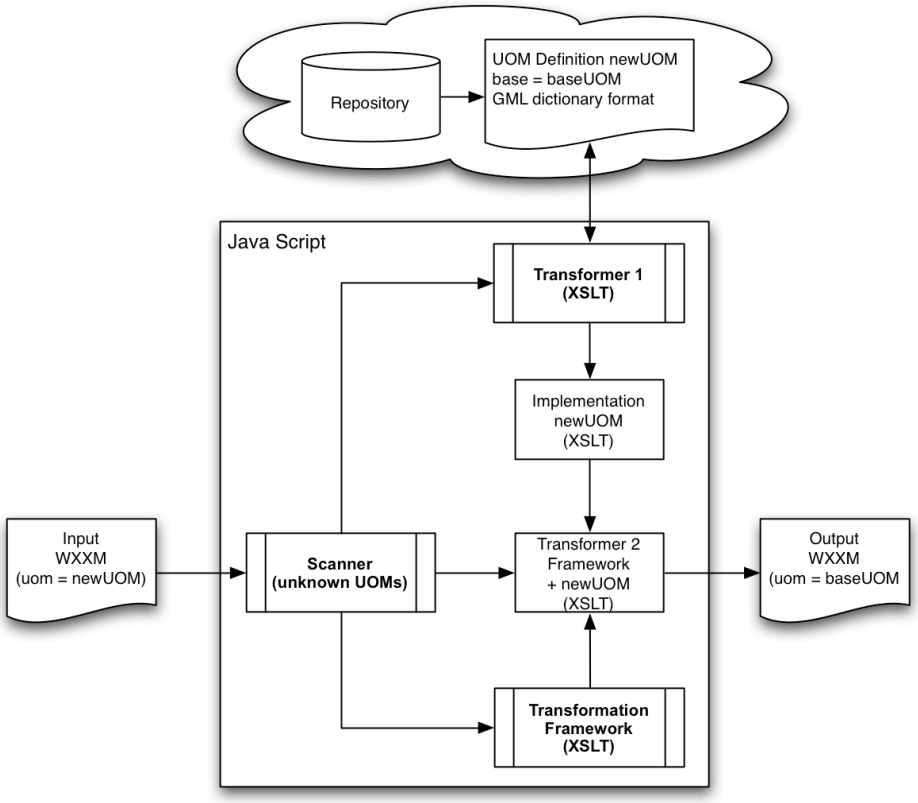


**Figure 25: Process implementation**

An *Input XML* file (here WXXM, see below) is scanned for unknown UOMs by a *Scanner Process*. This process refers to a profile outlining known UOMs. *Transformer 1* is only used in cases where unknown, new UOMs are found in the input file. If all UOMs used in the input file are known to the system, nothing happens and the output file is identical to the input file.

```
<avwx:AerodromeWx gml:id="KCRW_04">
    <avwx:windDirection uom="deg">0</avwx:windDirection>
    <avwx:windSpeed uom="kt">0</avwx:windSpeed>
    <avwx:cloudCondition>
        <wx:CloudCondition gml:id="KCRW_05">
            <wx:base uom="ogs.cs.latrobe.edu.au/newUOM">10500</wx:base>
                <!-- used to be 25000 ft -> * 0.42 (fictional new UOM) -->
            <wx:cloudAmount>SCATTERED</wx:cloudAmount>
        </wx:CloudCondition>
    </avwx:cloudCondition>
</avwx:AerodromeWx>
```

Should the *Scanner Process* detect unknown UOMs, it will follow the specified URL to read the definition of that unknown UOM from the remotely accessible repository (see below). This definition is implementation independent. Here, as explained previously, the GML dictionary was used to maintain consistency with the available data set for the experiment, however other methodologies, such as UCUM, would also be possible.

 

```
<gml:Dictionary gml:id="ID_1"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gml ./gml.xsd">
    <gml:name/>
    <gml:dictionaryEntry>
      <gml:ConventionalUnit gml:id="newUOM">
            <gml:name>new UOM</gml:name>
            <gml:quantityType>distance</gml:quantityType>
            <gml:catalogSymbol>nU</gml:catalogSymbol>
            <gml:conversionToPreferredUnit uom="#ft">
                  <gml:factor>0.42</gml:factor>
            </gml:conversionToPreferredUnit>
      </gml:ConventionalUnit>
    </gml:dictionaryEntry>
</gml:Dictionary>
```

It is now necessary to create an actual transformation implementation of the new UOM. In this example, XSLT is used to achieve this (*Transformer 1*, see below). *Transformer 1* uses the UOM definition as input and creates an XSLT template for this specific UOM. The template will transform an element with a UOM attribute with the unknown UOM as its value. It incorporates the *gml:factor* information as well as the *gml: conversionToPreferredUnit* value. In a similar fashion, *gml:formula* with its up to four child nodes corresponding to y = (a + bx) / (c + dx) could be facilitated instead of the *gml:factor.*

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:fn="http://www.w3.org/2005/xpath-functions"
xmlns:gml="http://www.opengis.net/gml" >
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

    <xsl:template match="/gml:Dictionary/gml:dictionaryEntry/gml:ConventionalUnit">
      <xsl:text disable-output-escaping="yes"><![CDATA[<xsl:template match="node()[@uom=']]></xsl:text>
      <xsl:value-of select="gml:catalogSymbol"/>
      <xsl:text disable-output-escaping="yes"><![CDATA[']">]]></xsl:text>

      <xsl:text disable-output-escaping="yes"><![CDATA[<xsl:variable name="factor" select="]]></xsl:text>
      <xsl:value-of select="gml:conversionToPreferredUnit/gml:factor"/>
      <xsl:text disable-output-escaping="yes"><![CDATA["/>]]></xsl:text>

      <xsl:text disable-output-escaping="yes"><![CDATA[<xsl:element name="{name()}">]]></xsl:text>
      <xsl:text disable-output-escaping="yes"><![CDATA[<xsl:attribute name="uom">]]></xsl:text>
      <xsl:value-of select="substring(gml:conversionToPreferredUnit/@uom,2)"/>
      <xsl:text disable-output-escaping="yes"><![CDATA[</xsl:attribute>]]></xsl:text>
      <xsl:text disable-output-escaping="yes"><![CDATA[<xsl:value-of select=". div $factor"/>]]></xsl:text>
      <xsl:text disable-output-escaping="yes"><![CDATA[</xsl:element>]]></xsl:text>

      <xsl:text disable-output-escaping="yes"><![CDATA[</xsl:template>]]></xsl:text>
    </xsl:template>

</xsl:stylesheet>
```

The resulting XSLT template is created as output (see below). This happens dynamically and as a result of the *UOM definition.*

```
<xsl:template match="node()[@uom='ogs.cs.latrobe.edu.au/newUOM']">
      <xsl:variable name="factor" select="0.128016"/>

      <xsl:element name="{name()}">
            <xsl:attribute name="uom">m</xsl:attribute>
            <xsl:value-of select=". div $factor"/>
      </xsl:element>
```

 

```
</xsl:template>
```

Together with an *XSLT Framework* which only copies input to output (for all those known UOMs and elements without the UOM attribute) this forms *Transformer 2* (see *XSLT Framework* below).

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:wx="http://www.eurocontrol.int/wx/1.1">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

    <xsl:template match="@*|node()">
      <xsl:copy>
            <xsl:apply-templates select="@*|node()"/>
      </xsl:copy>
    </xsl:template>

</xsl:stylesheet>
```

When applied to the original *Input WXXM* file, it transforms the new UOM to its base UOM while copying everything else to the output without change. The resulting *Output WXXM* looks like this:

```
<avwx:AerodromeWx gml:id="KCRW_04">
    <avwx:windDirection uom="deg">0</avwx:windDirection>
    <avwx:windSpeed uom="kt">0</avwx:windSpeed>
    <avwx:cloudCondition>
            <wx:CloudCondition gml:id="KCRW_05">
                  <wx:base uom="ft">25000</wx:base>
                  <wx:cloudAmount>SCATTERED</wx:cloudAmount>
            </wx:CloudCondition>
    </avwx:cloudCondition>
</avwx:AerodromeWx>
```

A possible implementation of this process using Java Script and XSLT can be seen in Figure 26. Any valid XML data can be copied into the top text area and transformed to local standards as long as *"uom"* attributes provide the new UOM's definition via a URL. Flexible profiles will be added in a future implementation.

 

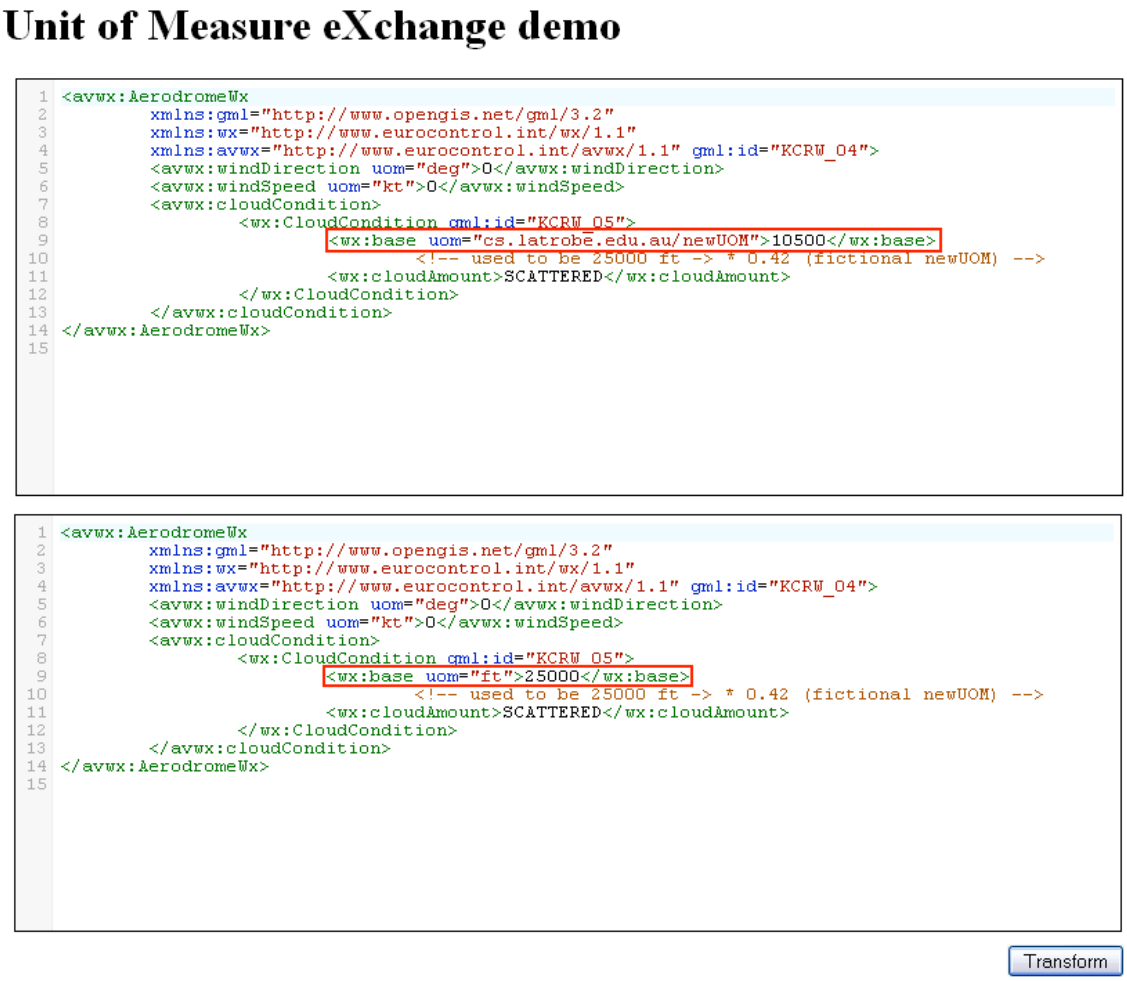


**Figure 26: Web Implementation**

### 6.2.4 Issues and recommendations

This section discusses the issues found and possible recommendations. It compares the different architectures, presents thoughts about performance and localization as well as the use of UOM profiles.

#### 6.2.4.1 Comparison

Table 5 compares the different architectures discussed earlier.

Centralized and de-centralized architecture, for the most part, are mutually exclusive in their strengths and weaknesses. Both inflict increased network traffic and thereby reduce performance.

 

Local Copy inherits the shortcomings of the centralized approach but would perform better by reducing network traffic. It is important to keep in mind that reliability and security can only be achieved if updates are properly managed.

Local Transformation, on the other hand, outperforms the previous approaches in terms of network traffic and performance. However, as it restricts encoding to the known standard only, it lacks flexibility and support for the new UOM in the long term.

**Table 5: Comparison of different UOM architectures**

| | Centralized | De-Centralized | Local Copy | Local Transform |
|---|---|---|---|---|
| Network traffic | - | - | + | + |
| Registration process | - | + | - | - |
| Growing repository | - | + | - | - |
| Political issues | - | + | - | - |
| Availability | + | - | + | + |
| Reliability | + | - | +[8] | +[8] |
| Security | + | - | +[9] | +[9] |
| Performance | - | - | + | + |

In our implementation approach, we use the local copy architecture combined with a de-centralized repository. This provides a highly flexible system while catering for network traffic and performance. Certainly, availability, reliability and security are challenges which need further discussion in this scenario.

#### 6.2.4.2 Performance: what is desired, what is to be expected?

Performance will be a critical criterion in a real-time environment such as in-flight updates. After speaking to industry shareholders, it became clear that XML data with its tag overhead might be a problem in itself. An online connection to transform data "on the fly" seems to be out of the question for some applications.

It is considered beneficial to conduct a series of real-life benchmarks and surveys to obtain a better understanding of current performance, possible performance and required minimum performance.

[8] If updates are managed.

[9] If the source can be trusted, the transformation can be trusted.

 

#### 6.2.4.3 Local versus remote access

Remote access raises the connectivity issue. It might be complicated to ensure that an application can, at any time, access a remote service via a network. Local services might not always be available and could be compromised by a third party.

The Local Transformation takes remote or local services out of the equation and replaces them with a local transformation plus a de-facto UOM standard.

#### 6.2.4.4 Version Management

With the architecture proposed in section 6.2.3 on page 32, only the definition of a new UOM is stored in a centralized or decentralized repository. The actual implementation, on the other hand, remains under the control of the user of the data. This raises the question of how to push new versions of a UOM definition to the shareholders. This will need further discussion should the approach with separated definition and implementation be applied. One possibility would be to include a version number in the UOM attribute's value. A new version would be considered a new UOM all together which requires a new implementation. Different versions could co-exist.

#### 6.2.4.5 Open vs. Secure

There may not be "one solution that fits all" for dealing with the distribution of UOM. Instead of either a very open and flexible **or** very secure system, a level-based implementation for UOM distribution, depending on the application environment could be considered. A participant could be certified for certain levels.

The levels proposed are:

- Level 0 - Zero restrictions: new UOM definitions and implementations can be hosted locally, without a registration process.
- Level 1 - Centralized repository for UOM definitions; new UOMs can only be added via a stringent verification process.
- Level 2 - Only known/standard UOMs are shared; new UOMs are kept only locally (e.g. in a research institution); they are converted locally into the standard UOMs (using a conversion function hosted by the owner of the UOMs) before sharing the data.
- Level 3 - Highest restriction: no new UOMs are allowed as long as "the shareholders" have not agreed; the set of UOMs forms a de-facto standard which is very secure and very fast, but also very inflexible.

 

### 6.3 Coordinate Reference System (CRS)

The following sections have been extracted from:

- https://portal.opengeospatial.org/twiki/bin/view/OWS8/AviationWeatherCrs
- OGC Web Coverage Service specification v 2.0
  http://www.opengeospatial.org/standards/wcs
- WCS Core change request CR
  https://portal.opengeospatial.org/files/?artifact_id=44308
- Weather Exchange Information Model WXXM
  http://www.eurocontrol.int/aim/public/standard_page/met_wie.html
- Encoding Representative Weather Datasets
  https://portal.opengeospatial.org/files/?artifact_id=46115

#### 6.3.1 Overview

The Coordinate Reference System (CRS) Domain Working Group develops strategies for encoding the earth coordinate reference systems and transformations between coordinate reference systems. The group has worked consistently on the joint OGC-ISO Document 19111 - Spatial Referencing by Coordinates.

The Aviation Thread of OWS-8 will explore ways to address the following two issues:

- How can a WCS client **request** a specific Coordinate Reference System (CRS) (including projection, projection parameters, datum, etc) when there is no existing EPSG (or other) code associated with the CRS? How can such CRSs be **encoded** in this case (well-known text, GML, etc)?

- Can a WCS 2.0 server formally advertise the capability to support **arbitrary** CRSs (as opposed to a fixed number of possible CRSs), as specified in the client query?

As a background, in the weather domain, the set of projected spaces of interest to clients is potentially infinite. For example, a data set available in a Lambert Conformal projection (with a certain projection origin and other parameterizations) may be requested by a client in Lambert Azimuthal Equal Area projection with a completely different origin and other parameterization. EPSG codes have been insufficient to describe the combinations of projections and projection parameters that the clients may require.

From the WCS server perspective, the requirement becomes that of supporting an infinite number of actual CRSs on-demand, and to be able to advertise such capability in their capabilities statement. Note the focus here is on solutions based on the WCS standard and not on the Web Processing Service (WPS), which could also be used to support custom CRS conversions.

### 6.3.2 Design Considerations

The proposed method for referencing CRSs is to use URL.

The issue here is that not all known standardized CRSs are URLified (not even all EPSG codes, not to speak of climate community 4D CRSs etc.). Task here seems to be to **collect**, **unify**, and **define** URLs.

The challenging task here is that the advanced functionality as offered, e.g., by WCS requires a **dynamic** axis combination into a new CRS. As these are generated on the fly, as a side effect of coverage processing, they cannot be predefined, hence a client cannot know them in advance. Further, the sheer combinatorial complexity prevents from enumerating them as URLs. So we need a mechanism to **transport** CRS definitions.

Specific core issues are:

- Non-standard axis **combination**. By "non-standard" mainly EPSG is meant. Example: 4D data need a combination of x/y (EPSG!), z, and t (ISO 8601!) axes; x/t slices from such a 4D cube need to return a coverage having an x/t based CRS.

- **Parametric** CRSs. Example: Polar stereographic CRSs can have a reference angle of 0 (Greenwich) or any other angle.

- "**Abstract**" axes: axes having no spatio-temporal semantics, such as "extra" time axes, pressure, products offered, etc.

### 6.3.3 Deployment

Following the evaluation of different approaches, the following set of work items has been suggested for the deployment:

1. WCS Core change request CR: adding a CRS GetCoverage request parameter and provision of CRSs supported in the Capabilities. The Change Request can be found at: https://portal.opengeospatial.org/files/?artifact_id=44308

2. Complete replacement of 07-092r3 through the creation of an OGC-NA Name Type Specification (NTS) for predefined, combined, and parameterized CRSs to augment the /def/ NTS (which defines http URIs for standard CRS).

3. Devise AUTO CRS definitions.

4. Specification of CRS registry service (cf. WCTS).

5. Change request CR to GML: EnvelopeWithCRSType (similar to EnvelopeWithTimePeriodType).

 

### 6.3.4 Issues and Recommendations

Aaron Braeckel has raised the issue of Client/server differentiation:

- How would clients request a **specific** CRS (including projection, projection parameters, datum, etc.) where there is no existing EPSG (or other) code that covers it?
- How would servers formally advertise the capability to produce **arbitrary** CRSs?

Other issues are:

**Issue 1:** How to represent ad-hoc **combined** CRSs in a request capability document etc.?

Following different considerations including the use of URNs, Well-Known Text, URLs, and GML, the recommendation is to establish a registry service resolving complete, combined, and parameterized CRS URLs. This service would return "flat", non-parameterized GML definitions per invocation (i.e., with good luck we can reuse GML as is).

**Issue 2:** How to inform clients about CRS support?

Description: A client needs to know in which CRS it can request coverage. To this end, the identifying URLs need to be provided by a server.

Alternatives:

- Alternative 1: associate CRSs with the service (ServiceMetadata) and deliver as part of the capabilities.
- Alternative 2: associate CRSs with individual coverages (ServiceParameters) and deliver as part of a coverage description.

As delivering information only in the service, not the coverage description saves volume, and since ServiceMetadata is simpler for the client to use, it seems that the use of ServiceMetadata is a better approach.

## 6.4 Web Coverage Services (WCS)

The following sections have been extracted from:

- http://www.opengeospatial.org/standards/wcs
- https://portal.opengeospatial.org/twiki/bin/view/OWS8/JacobsWCS

- WCS Core change request CR
  https://portal.opengeospatial.org/files/?artifact_id=44308
- http://geobrain.laits.gmu.edu/ows8/aviationWCS.html
- Encoding Representative Weather Datasets
  https://portal.opengeospatial.org/files/?artifact_id=46115

### 6.4.1 Overview

The OpenGIS® Web Coverage Service Interface Standard (WCS) defines a standard interface and operations that enables interoperable access to geospatial "coverages" [http://www.opengeospatial.org/ogc/glossary/c]. Coverage is a feature that associates positions within a bounded space (its spatiotemporal domain) to feature attribute values (its range). GIS coverages (including the special case of Earth images) are two- (and sometimes higher-) dimensional metaphors for phenomena found on or near a portion of the Earth's surface. A coverage can consist of a set of features or Feature Collections. Earth images are seen as Grid Coverages that contain features whose geometries are of the type "set of cells" or "set of pixels" (surfaces).

The term "grid coverages" typically refers to content such as satellite images, digital aerial photos, digital elevation data, and other phenomena represented by values at each measurement point.

Radar data is sometimes represented in more than 4D representation, such as when multiple radars make observations at the same physical point. The requirement is to recommend ways for representing such 5D data as a WCS coverage.

The analysis of the problem statement concluded that the issue is entirely one of encoding and extending the WCS specification to include governance over content data is significantly outside the scope of the specification.

### 6.4.2 Operations

The web coverage service supports the following operations:

- WCS GetCapabilities
- WCS DescribeCoverage
- WCS GetCoverage
- WCPS ProcessCoverages

 

Data sets (both pre-existing ones and those specific to OWS-8, which will be announced upon availability) are accessible on all service endpoints simultaneously. This means, among other things, that a "GetCapabilities" for WCS returns information pertinent not only to a WCS "GetCoverage" request, but also to WCPS "ProcessCoverages" and a WPS "Execute". Supported protocols are GET/KVP and XML/POST for WCS; XML/POST for WCPS; and GET/KVP for WPS.

#### 6.4.2.1 GetCapabilities

```
service endpoint: http://212.201.49.173:8080/earthlook-ras
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCapabilities  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:wcs="http://www.opengis.net/wcs/2.0"
   xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd" service="WCS">
   <ows:AcceptVersions>
      <ows:Version>2.0.0</ows:Version>
   </ows:AcceptVersions>
</wcs:GetCapabilities>
```

#### 6.4.2.2 DescribeCoverage

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:DescribeCoverage  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:wcs="http://www.opengis.net/wcs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd"
   service="WCS" version="2.0.0">
   <wcs:CoverageId>climate_clouds</wcs:CoverageId>
</wcs:DescribeCoverage>
```

#### 6.4.2.3 GetCoverage 2D

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCoverage xmlns:wcs="http://www.opengis.net/wcs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd"
   service="WCS" version="2.0.0">
   <wcs:CoverageId>NIR</wcs:CoverageId>
   <wcs:DimensionTrim>
      <wcs:Dimension>x</wcs:Dimension>
      <wcs:TrimLow>20</wcs:TrimLow>
      <wcs:TrimHigh>29</wcs:TrimHigh>
   </wcs:DimensionTrim>
   <wcs:DimensionSlice>
      <wcs:Dimension>y</wcs:Dimension>
      <wcs:SlicePoint>1</wcs:SlicePoint>
   </wcs:DimensionSlice>
</wcs:GetCoverage>
```

#### 6.4.2.4 GetCoverage 3D

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCoverage xmlns:wcs="http://www.opengis.net/wcs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
```

 

```
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd"
   service="WCS" version="2.0.0">
   <wcs:CoverageId>climate_clouds</wcs:CoverageId>
   <wcs:DimensionTrim>
      <wcs:Dimension>x</wcs:Dimension>
      <wcs:TrimLow>20</wcs:TrimLow>
      <wcs:TrimHigh>29</wcs:TrimHigh>
   </wcs:DimensionTrim>
   <wcs:DimensionTrim>
      <wcs:Dimension>y</wcs:Dimension>
      <wcs:TrimLow>20</wcs:TrimLow>
      <wcs:TrimHigh>29</wcs:TrimHigh>
   </wcs:DimensionTrim>
   <wcs:DimensionSlice>
      <wcs:Dimension>t</wcs:Dimension>
      <wcs:SlicePoint>3</wcs:SlicePoint>
   </wcs:DimensionSlice>
</wcs:GetCoverage>
```

#### 6.4.3 CSISS-GMU Deployment

The following section reports the deployment from CSISS-GMU for serving NOAA GOES Wind products.

WCS supports the electronic retrieval of geospatial data as coverages which are digital geospatial information representing space/time-varying phenomena.
Implementation Version: 2.0.0 (OGC 09-110r3)
Support protocol: HTTP GET (OGC 09-147r1)

NOAA GOES Wind products - High Density Satellite Derived Winds.
Each record stores four fields: air_temperature, air_pressure, wind_direction and wind_speed.

Reference: http://www.ssd.noaa.gov/PS/WIND/

##### 6.4.3.1 WCS deployed operations

##### 6.4.3.2 GetCapabilities

The mandatory GetCapabilities operation allows WCS clients to retrieve service metadata and coverage brief description from a server. The response to a GetCapabilities request shall be an XML document containing service metadata about the server and the coverage information.

GetCapabilities request URL:
http://geobrain.laits.gmu.edu/ows8aviation/goeswindwcs?service=wcs&version=2.0.0&request=getcapabilities

#### 6.4.3.3 DescribeCoverage

The client will need to issue a DescribeCoverage request to obtain a full description of one or more coverages available. The server responds to such a request with an XML document describing one or more coverages served by the WCS.

DescribeCoverage request URL:
http://geobrain.laits.gmu.edu/ows8aviation/goeswindwcs?service=wcs&version=2.0.0&request=describecoverage&coverageid=GOES-WEST-201110509Z:Cloud-Drift-Wind

#### 6.4.3.4 GetCoverage

The GetCoverage operation allows the retrieval of coverages from a coverage offering. The WCS server processes a GetCoverage request and returns a response file to the client. For GOES Wind data, KML and WXXM format are supported. This WCS also supports the rangeSubset, where the client can select a range of interest by specifying its corresponding field name. If the rangeSubset parameter is not specified, all fields will be returned.

GetCoverage request URL 1 (Without rangesubset):
http://geobrain.laits.gmu.edu/ows8aviation/goeswindwcs?service=wcs&version=2.0.0&request=getcoverage&coverageid=GOES-WEST-201110509Z:Cloud-Drift-Wind&subset=Lat,http://www.opengis.net/def/crs/EPSG/0/4326(22,24)&subset=Lon,http://www.opengis.net/def/crs/EPSG/0/4326(-158,-156)&format=wxxm

GetCoverage request URL 2 (With rangesubset, select wind_speed field):
http://geobrain.laits.gmu.edu/ows8aviation/goeswindwcs?service=wcs&version=2.0.0&request=getcoverage&coverageid=GOES-WEST-201110509Z:Cloud-Drift-Wind&subset=Lat,http://www.opengis.net/def/crs/EPSG/0/4326(22,24)&subset=Lon,http://www.opengis.net/def/crs/EPSG/0/4326(-158,-156)&format=wxxm&rangesubset=wind_speed

GetCoverage request URL 3 (KML output):
http://geobrain.laits.gmu.edu/ows8aviation/goeswindwcs?service=wcs&version=2.0.0&request=getcoverage&coverageid=GOES-WEST-201110509Z:Cloud-Drift-Wind&subset=Lat,http://www.opengis.net/def/crs/EPSG/0/4326(22,24)&subset=Lon,http://www.opengis.net/def/crs/EPSG/0/4326(-158,-156)&format=kml

### 6.4.4 Issues and Recommendation

- WXXM ingest
- WXXM output
- Sample requests
- Orchestration with data providers and consumers

#### 6.4.5 Jacobs University | rasdaman GmbH WCS & WCPS & WPS Deployment

In the course of OWS-8, a series of multi-dimensional data sets of various origins has been imported into the PostgreSQL / rasdaman database. Datasets are accessible on all service endpoints, simultaneously. This means, among other things, that a "GetCapabilities" for WCS returns information pertinent not only to a WCS "GetCoverage" request, but also relevant to a WCPS "ProcessCoverages" and a WPS "Execute". Supported protocols are GET/KVP and XML/POST for WCS; XML/POST for WCPS; and GET/KVP for WPS.

**Acknowledgement:** Data import mostly has been accomplished by Michael Owonibi. Implementation in OWS-8 mainly has been done by Dimitar Misev.

##### 6.4.5.1 By Scenario

WXXM (Aviation Thread):

- Data:Temperature, u_wind, v_wind, geopotential_height
- Demo Queries:
  - Slicing ( submit query ) : retrieve 2D data from a 5D temperature data by slicing along 3 dimensions
  - Absolute wind speed computation ( submit query ) : absolute wind speed from vertical and horizontal wind components of a 5D data
  - See below for more WCPS demo sample queries

Amazon Drought (Observation Fusion Thread):

- Data: EVI, NDVI, Land Cover Classification, Quality Mask (pre-materialized) Rainfall(TRMM)
- Prematerialized Data: Reference Average Rainfall, NDVI, EVI from 2000 to 2009, Reference Standard Deviation of Rainfall, NDVI, EVI from 2000 to 2009, Mean Rainfall, EVI, NDVI for 2010 and 2005
- WCPS demo queries on the original data:
  - Average JAS NDVI for year 2000 ( submit query ) : Derviation of the yearly average from the monthly average

```
for c in ( NDVI_AMAZON )
return
  encode(  (char)(((c[t(0)] + c[t(1)] + c[t(2)] ) / 3 ) / 40 ) , "png")
```

 

- WCPS demo queries on pre-materialized data:
  - Area experiencing drought in 2005 ( submit query ) : Computed from mean rainfall 2005, reference mean rainfall and reference standard deviation of rainfall

```
for c in ( RAINFALL_2005_AMAZON ),
    d in ( RAINFALL_REF_AVG_AMAZON ),
    e in ( RAINFALL_REF_SD_AMAZON )
return
  encode(  ( ( (c - d) / e ) > (0) ) , "png")
```

  - computation and classification of the anomaly of the Amazon drought area
  - see below for sample WCPS queries.
- Oil Spill (Observation Fusion Thread):
  - Data :Oil Spill Time Series, Coast Map
  - WCPS demo Queries :
    - Slicing
    - overall contamination computation
    - time diagram with summary data

#### 6.4.5.2 By Dataset

**Temperature_5D, Geopotential_height_5D, u_wind_5D, v_wind_5D**

- thread: aviation
- bounding box Monterey Bay: Longitude(-122.53,-121.45), Latitude (36.24,37.06)
- data origin: http://www.ral.ucar.edu/staff/braeckel/ows8/ruc-model/
- data description: These are 5D with axis names modelTime, pressure, t, x, y and dimension [2,11,18,151,131].
- service links: http://212.201.49.173:8080/ows8
- sample WCPS query:

```
for c in ( Temperature_5D )
return
 encode( (char)((c[ t(0), modelTime(0), pressure(0) ] - 240) * 4), "png" )
```

- sample WCPS-WPS GetKVP requests (retrieval of 2D data from 5D data by Slicing in along 3 dimensions):

```
Execute Temperature_5D Slicing
Execute u_wind_5D Slicing
Execute v_wind_5D Slicing
```

- sample WCS XML requests:

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCoverage xmlns:wcs=http://www.opengis.net/wcs/2.0
  xmlns:gml=http://www.opengis.net/gml/3.2
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:schemaLocation=http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd
  service="WCS" version="2.0.0">
  <wcs:CoverageId>u_wind_5D</wcs:CoverageId>
  <wcs:DimensionTrim>
    <wcs:Dimension>x</wcs:Dimension>
    <wcs:TrimLow>100</wcs:TrimLow>
    <wcs:TrimHigh>109</wcs:TrimHigh>
  </wcs:DimensionTrim>
  <wcs:DimensionTrim>
    <wcs:Dimension>y</wcs:Dimension>
    <wcs:TrimLow>100</wcs:TrimLow>
    <wcs:TrimHigh>109</wcs:TrimHigh>
  </wcs:DimensionTrim>
  <wcs:DimensionSlice>
    <wcs:Dimension>t</wcs:Dimension>
    <wcs:SlicePoint>3</wcs:SlicePoint>
  </wcs:DimensionSlice>
  <wcs:DimensionSlice>
    <wcs:Dimension>pressure</wcs:Dimension>
    <wcs:SlicePoint>3</wcs:SlicePoint>
  </wcs:DimensionSlice>
  <wcs:DimensionSlice>
    <wcs:Dimension>modelTime</wcs:Dimension>
    <wcs:SlicePoint>1</wcs:SlicePoint>
  </wcs:DimensionSlice>
</wcs:GetCoverage>
```

- sample WPS-WCPS PostXML request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
```

 

```
  <ows:Identifier>petascope.wps.n52.ProcessCoverages</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>Query</ows:Identifier>
      <wps:Data>
        <wps:ComplexData mimeType="text/plain">
        for c in ( Temperature_5D )
        return encode( (char)((c[ t(0), modelTime(0), pressure(0) ] - 240) * 4), "png" )
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput mimeType="image/png">
       <ows:Identifier>CoverageList</ows:Identifier>
    </wps:RawDataOutput>
  </wps:ResponseForm>
</wps:Execute>
```

**OIL_SPILL_MONTEREY_BAY**

- data origin: ESRI
- data description: 3D (x, y, t) monthly rainfall data with geo-extents given as being between -91.4718 and -86.0068 longitude and 27.51694 and 30.19194 latitude
- service links: kahlua.eecs.jacobs-university.de:8080/ows8
- sample WCPS query:

```
for    c in ( OIL_SPILL_MONTEREY_BAY  )
return
 encode( (char) (c[ t(0) ]  * 60)  , "png" )
```

- sample WCPS-WPS GetKVP? requests : Get the Data on the first day of the oil spill
- sample WCS request:

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCoverage xmlns:wcs=http://www.opengis.net/wcs/2.0
    xmlns:gml=http://www.opengis.net/gml/3.2
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd"
   service="WCS" version="2.0.0">
  <wcs:CoverageId>OIL_SPILL_MONTEREY_BAY </wcs:CoverageId>
  <wcs:DimensionSlice>
    <wcs:Dimension>t</wcs:Dimension>
    <wcs:SlicePoint>0</wcs:SlicePoint>
```

 

```
  </wcs:DimensionSlice>
</wcs:GetCoverage>
```

- sample WPS-WCPS POST XML requests:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>petascope.wps.n52.ProcessCoverages</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>Query</ows:Identifier>
      <wps:Data>
        <wps:ComplexData mimeType="text/plain">
        for c in ( OIL_SPILL_MONTEREY_BAY )
        return encode( (c[ t(0) ]  ), "png" )
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput mimeType="image/png">
      <ows:Identifier>CoverageList</ows:Identifier>
    </wps:RawDataOutput>
  </wps:ResponseForm>
</wps:Execute>
```

**TRMM (Rainfall Data)**

- data origin: http://mirador.gsfc.nasa.gov/cgi-bin/mirador/presentNavigation.pl?tree=project&project=TRMM&dataGroup=Gridded&dataset=3B43:%20Monthly%200.25%20x%200.25%20degree%20merged%20TRMM%20and%20other%20sources%20estimates&version=006

- data description: 3D (x, y, t) monthly rainfall data with geo-extents given as -50 - 50 degree latitude, and -180 - 180 degree longitude.More information on http://disc.sci.gsfc.nasa.gov/precipitation/documentation/TRMM_README/TRMM_3B43_readme.shtml. There is potential of using the data in Amazon drought scenario

- service links: http://212.201.49.173:8080/ows8

- sample WCPS query:

```
for  c in ( TRMM )
return
 encode( c[ t(0) ] , "png" )
```

- sample WCPS-WPS GetKVP request: “Get rainfall map over Amazon area in July, 2000”:

```
http://212.201.49.173:8080/ows8/wps?service=WPS&Version=1.0.0&Request=Execute&identifier=ProcessCoverages&DataInputs=[WcpsAbstractSyntax=for%20c%20in%20(%20RAINFALL_TRMM_AMAZON)%20return%20encode(%20c[%20t(0)%20]%20%20,%20%22png%22%20)]
```

- sample WCS request:

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCoverage xmlns:wcs="http://www.opengis.net/wcs/2.0"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd" service="WCS"
version="2.0.0">
  <wcs:CoverageId>TRMM</wcs:CoverageId>
  <wcs:DimensionSlice>
    <wcs:Dimension>t</wcs:Dimension>
    <wcs:SlicePoint>0</wcs:SlicePoint>
  </wcs:DimensionSlice>
</wcs:GetCoverage>
```

- sample WPS-WCPS POST XML requests:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>petascope.wps.n52.ProcessCoverages</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>Query</ows:Identifier>
      <wps:Data>
        <wps:ComplexData mimeType="text/plain">
        for c in ( TRMM )
        return encode( c[ t(0) ] , "png" )
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput mimeType="image/png">
       <ows:Identifier>CoverageList</ows:Identifier>
    </wps:RawDataOutput>
```

 

```
  </wps:ResponseForm>
</wps:Execute>
```

**NDVI_AMAZON, EVI_AMAZON, VI_Quality_AMAZON**

- data origin: ftp://e4ftl01.cr.usgs.gov/MOLT/MOD13A2.005/
- data description: 3D (x, y, t) data with a 16-day temporal resolution. Only data for July, August and September from 2000 – 2010 will be used in the Amazon drought use case scenario.
- service links: kahlua.eecs.jacobs-university.de:8080/ows8
- sample WCPS request:

```
for    c in ( NDVI_AMAZON )
return
 encode( c[ t(0) ] , "png" )
```

- sample WCPS-WPS Get KVP requests: “Get the NDVI over Amazon area in July, 2000”:

```
http://212.201.49.173:8080/ows8/wps?service=WPS&Version=1.0.0&Request=Execute&identifi
er=ProcessCoverages&DataInputs=[WcpsAbstractSyntax=for%20c%20in%20(%20NDVI_AMAZON
)%20return%20encode(%20c[%20t(0)%20]%20%20,%20%22png%22%20)]
```

- sample WCS requests:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <wcs:GetCoverage xmlns:wcs="http://www.opengis.net/wcs/2.0"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd" service="WCS"
version="2.0.0">
    <wcs:CoverageId>NDVI_AMAZON</wcs:CoverageId>
    <wcs:DimensionSlice>
    <wcs:Dimension>t</wcs:Dimension>
    <wcs:SlicePoint>0</wcs:SlicePoint>
    </wcs:DimensionSlice>
    </wcs:GetCoverage>
```

**radar_base_reflectivity**

- data origin: http://www.ral.ucar.edu/staff/braeckel/ows8/radar-dbz/
- data description: 3D (x, y, t)

 

- service links: http://212.201.49.173:8080/ows8
- sample WCPS request:

```
for    c in (  radar_base_reflectivity )
return
  encode( c[ t(0) ] , "png" )
```

- sample WCS request:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <wcs:GetCoverage xmlns:wcs="http://www.opengis.net/wcs/2.0"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd" service="WCS"
version="2.0.0">
    <wcs:CoverageId> radar_base_reflectivity</wcs:CoverageId>
    <wcs:DimensionSlice>
    <wcs:Dimension>t</wcs:Dimension>
    <wcs:SlicePoint>0</wcs:SlicePoint>
    </wcs:DimensionSlice>
    </wcs:GetCoverage>
```

- sample WPS-WCPS requests ( execute here):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>petascope.wps.n52.ProcessCoverages</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>Query</ows:Identifier>
      <wps:Data>
        <wps:ComplexData mimeType="text/plain">
        for c in ( radar_base_reflectivity )
        return encode( c[ t(0) ] , "png" )
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput mimeType="image/png">
       <ows:Identifier>CoverageList</ows:Identifier>
    </wps:RawDataOutput>
  </wps:ResponseForm>
</wps:Execute>
```

### 6.4.6 By Service

**WCS 2.0**

- service endpoint: http://212.201.49.173:8080/ows8
- formats supported: GML (use WCPS or WPS below if other formats are required)
- GetCapabilities:

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCapabilities  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ows="http://www.opengis.net/ows/2.0"
   xmlns:wcs="http://www.opengis.net/wcs/2.0"
   xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd" service="WCS">
   <ows:AcceptVersions>
      <ows:Version>2.0.0</ows:Version>
   </ows:AcceptVersions>
</wcs:GetCapabilities>
```

- DescribeCoverage:

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:DescribeCoverage  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:wcs="http://www.opengis.net/wcs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd"
   service="WCS" version="2.0.0">
   <wcs:CoverageId>TRMM</wcs:CoverageId>
</wcs:DescribeCoverage>
```

- GetCoverage 2D:

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCoverage xmlns:wcs="http://www.opengis.net/wcs/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.2"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd"
   service="WCS" version="2.0.0">
   <wcs:CoverageId>LAND_USE_AMAZON</wcs:CoverageId>
   <wcs:DimensionTrim>
      <wcs:Dimension>x</wcs:Dimension>
      <wcs:TrimLow>20</wcs:TrimLow>
      <wcs:TrimHigh>29</wcs:TrimHigh>
   </wcs:DimensionTrim>
   <wcs:DimensionSlice>
      <wcs:Dimension>y</wcs:Dimension>
```

 

```
    <wcs:SlicePoint>1</wcs:SlicePoint>
  </wcs:DimensionSlice>
</wcs:GetCoverage>
```

- GetCoverage 3D:

```
<?xml version="1.0" encoding="UTF-8"?>
<wcs:GetCoverage xmlns:wcs="http://www.opengis.net/wcs/2.0"
  xmlns:gml="http://www.opengis.net/gml/3.2"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wcs/2.0 ../../wcsAll.xsd"
  service="WCS" version="2.0.0">
  <wcs:CoverageId>radar_base_reflectivity</wcs:CoverageId>
  <wcs:DimensionTrim>
    <wcs:Dimension>x</wcs:Dimension>
    <wcs:TrimLow>740</wcs:TrimLow>
    <wcs:TrimHigh>749</wcs:TrimHigh>
  </wcs:DimensionTrim>
  <wcs:DimensionTrim>
    <wcs:Dimension>y</wcs:Dimension>
    <wcs:TrimLow>880</wcs:TrimLow>
    <wcs:TrimHigh>889</wcs:TrimHigh>
  </wcs:DimensionTrim>
  <wcs:DimensionSlice>
    <wcs:Dimension>t</wcs:Dimension>
    <wcs:SlicePoint>3</wcs:SlicePoint>
  </wcs:DimensionSlice>
</wcs:GetCoverage>
```

**WCPS 1.0**

- service endpoint: http://212.201.49.173:8080/ows8
- formats supported: TIFF, BMP, JPEG, PNG, CSV, NetCDF? , HDF4, ...
- ProcessCoverages:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ProcessCoveragesRequest xmlns="http://www.opengis.net/wcps/1.0" service="WCPS"
version="1.0.0">
  <query>
    <abstractSyntax>
        for c in ( OIL_SPILL_MONTEREY_BAY )
        return encode( (c[ t(0) ]  ), "png" )
    </abstractSyntax>
  </query>
</ProcessCoveragesRequest>
```

**WPS 1.0**

- service endpoint: (currently, only GET KVP requests are available )
- formats supported: TIFF, BMP, JPEG, PNG, CSV, NetCDF? , HDF4, ...
- WPS GetCapabilities:

```
http://212.201.49.173:8080/earthlook-ras/wps?service=WPS&Version=1.0.0&Request=GetCapabilities
```

- WPS DescribeProcess:

```
http://212.201.49.173:8080/earthlook-ras/wps?service=WPS&Request=DescribeProcess&Version=1.0.0&Identifier=ProcessCoverages
```

- WPS Execute:

```
http://212.201.49.173:8080/ows8/wps?service=WPS&Version=1.0.0&Request=Execute&identifier=ProcessCoverages&DataInputs=[WcpsAbstractSyntax=for c in ( NDVI_AMAZON) return encode( c[ t(0) ] , "png" )]
```

 