

# Open Geospatial Consortium, Inc.

Date: 2008-08-22

Reference number of this document: [08-124](#)

Version: 0.5.0

Category: **OGC™** Engineering Report

Editor: Luis Bermudez

## **OGC® Ocean Science Interoperability Experiment Phase 1 Report**

Copyright© 2008 Open Geospatial Consortium. All Rights Reserved  
To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

### **Warning**

This document is not an OGC Standard. This document presents a discussion of technology issues considered in an initiative of the OGC Interoperability Program. This document does not represent an official position of the OGC. It is subject to change without notice and may not be referred to as an OGC Standard. However, the discussions in this document could very well lead to the definition of an OGC Standard. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

## **Preface**

This document provides lessons learned and best practices resulted from the Ocean Science Interoperability Experiment (Oceans IE). The Oceans IE was created to investigate the use of OGC Web Feature Services (WFS) and OGC Sensor Observation Services (SOS) for representing and exchanging point data records from fixed in-situ marine platforms.

<b>Contents</b>	<b>Page</b>
1 Introduction.....	2
1.1 Summary and Scope .....	2
1.2 Foreword.....	2
Document contributor contact points .....	3
Revision history .....	3
Future work.....	4
2 Normative References .....	4
3 Terms and definitions.....	4
4 Conventions .....	5
4.1 Abbreviated terms .....	5
4.2 UML Notation.....	5
5 Background.....	6
5.1 Motivation.....	6
5.2 User Scenario .....	7
5.3 Detailed Use Cases .....	7
6 Ocean Observation and Measurement.....	8
6.1 Definition .....	8
6.2 Query .....	8
7 OGC Comparison result between WFS and SOS .....	9
7.1 Web Feature Service.....	9
7.2 Sensor Observation Service .....	11
7.3 Comparison of getFeature (WFS) with GetObservation (SOS).....	12
7.3.1.1 Request Comparison .....	13
7.3.1.2 Response Comparison .....	14
8 Other Comparison Results.....	14
9 SOS Implementation Guide.....	16
9.1 Modeling Systems of Systems .....	16
<b>9.1.1</b> Observation .....	16
<b>9.1.2</b> Feature of Interest.....	16
<b>9.1.3</b> Procedure.....	16
<b>9.1.4</b> Result .....	16
9.2 Feature of Interest.....	17
9.3 Semantic Mediation and categories.....	19
9.4 Uniform Resource Identifiers.....	20
<b>9.4.1</b> How to use OGC URNs.....	21
<b>9.4.2</b> Using user-defined dictionaries.....	21
9.4.2.1 authority.....	22
9.4.2.2 version .....	22
9.4.2.3 resourceType, ontology name, and file name.....	22
9.4.2.4 fileExtension .....	24
9.4.2.5 Terms.....	24
9.5 HTTP Get Request .....	25
<b>9.5.1</b> Capitalization of Keyword Value Pair names and values.....	26

9.5.2 Escaping special characters.....	26
9.5.3 GetCapabilities Request.....	26
9.5.4 DescribeSensor Request.....	27
9.5.5 GetObservation Request .....	27
9.6 Vertical Datum Coordinate reference system .....	28
9.6.1 Sea level-based systems .....	29
9.6.2 GeoID.....	29
9.6.3 Bottom-Based Systems .....	29
9.6.4 Namespaces for Combined CRSs.....	30
9.6.5 GetLatest .....	30
10 Acknowledgements .....	31

<b>Figures</b>	Page
<b>Figure 1 – Partial View of GetFeature Request .....</b>	<b>10</b>
<b>Figure 2 – Observation and System model used in OOSTethys.....</b>	<b>17</b>
<b>Figure 3 – Categorization of observations at openioos.org .....</b>	<b>19</b>

<b>Tables</b>	Page
<b>Table 1 – Query Components.....</b>	<b>9</b>
<b>Table 2 – List of investigated WFS services.....</b>	<b>12</b>
<b>Table 3 – List of investigated SOS services .....</b>	<b>12</b>
<b>Table 4 – Query decomposition and default support by WFS and SOS.....</b>	<b>13</b>
<b>Table 5 – URL Structure.....</b>	<b>25</b>
<b>Table 6 – Reserved Characters .....</b>	<b>25</b>
<b>Table 7 – GetCapabilities request parameters .....</b>	<b>26</b>
<b>Table 8 – DescribeSensor Request Parameters .....</b>	<b>27</b>
<b>Table 9 – GetObservation Request Parameters .....</b>	<b>27</b>



# OpenGIS® Ocean Science Interoperability Experiment 1

## 1 Introduction

### 1.1 Summary and Scope

This document provides lessons learned and best practices resulted from the Ocean Science Interoperability Experiment (Oceans IE). Institutions involved at OOSTethys (SURA, UNIDATA, MMI, and OpenIOOS) initiated the Oceans IE. The experiment is seeking to advance standard best practices for publishing of marine observations.

The Oceans IE Phase I investigated the use of OGC Web Feature Services (WFS) and OGC Sensor Observation Services (SOS) for representing and exchanging point data records from fixed in-situ marine platforms. It concluded that the use of OGC Sensor Observation Services (SOS) was better suited than the use of OGC Web Feature Services (WFS) for this purpose. By publishing an SOS service instead of a WFS service communities will not required to create and maintain schemas, and interoperability at the client side is achieved; however this requires an effort in creating and maintaining controlled vocabularies by marine communities.

The Oceans IE developed the following best practices for using an OGC Sensor Observation Service (1.1), which will help improve existing standards and recommendations at OGC:

- Requesting a get latest observation
- Encoding of OGC URNs when versioning is missing
- Publishing of URIs by service providers
- Using Semantic Web technologies to categorize SOS services
- Publishing an SOS as an HTTP-Get service
- Encoding vertical datums (Sea level based systems, geoid based systems and bottom based systems) in marine observations

Also, the OOSTethys team, developed a set of toolkits to help service providers publish SOS services. The toolkits are in Java, Perl and Python and follow the best practices detailed in this document. The toolkits are available at <http://www.oostethys.org>.

### 1.2 Foreword

This is an informative document that describes lessons learned and best practices from using OGC, as well as, W3C standards. This is not an OGC or W3C standard.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The knowledge of the authors of this report, no such proprietary material is revealed in this document. The OGC shall not be held responsible

for identifying any or all such patent rights.

**Document contributor contact points**

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

Name	Organization
Luis Bermudez	Southeastern Universities Research Association
John Graybeal	Monterey Bay Aquarium Research Institute
Gerry Creager	Texas A&M University
Tony Cook	Texas A&M University
David Forrest	Virginia Institute of Marine Sciences
Philip Bogden	Gulf of Maine Ocean Observing System
Charlton Purvis	Texas A&M University
Eric Bridger	Gulf of Maine Ocean Observing System

**Revision history**

Date	Release	Editor	Primary clauses modified	Description
2008-03-17		JG	Throughout	Forma document in the OGC template
2008-06-26		LB	Throughout	Extensive editorial corrections
2008-07-09		LB	Throughout	Editorial corrections, and removed template comments
2008-07-17		LB	Section 1, Section 7	Added to future work in sec 1. Updated TAMU links. Major revisions section 9. Added 9.1 and 9.2
2008-07-31		PB	Section 5	Added section 5
2008-08-01		LB	Throughout	Extensive editorial corrections, added summary.
2008-08-17		LB	Section 9	Added best practices
2008-08-17		DF	Section 9	Added section about vertical datums 9.7
2008-08-18		GC	Throughout	Extensive editorial corrections
2008-08-19		LB	Section 9	Editorial corrections, added best practices about URI construction, reorganized section 9, cleaned sections 2-4
2008-08-19		JG	Section 9	Editorial corrections. Section 9
2008-08-20		LB	Section 9	Editorial corrections. Section 9
2008-08-20		JG	Section 9	Editorial corrections. Section 9
2008-08-21		LB	Throughout	Editorial corrections.
2008-08-22		LB	Section 10	Added.

## Future work

Improvements in this document are desirable to amplify details of the specifications and resources used within the OGC standards (for example, vocabulary term resources).

The Ocean Science Interoperability Experiment will continue to several phases. Possible topics will include:

- Investigate more complex observations and features in the marine domain.
- Investigate existing marine sampling features models and SWE conceptual models (e.g O&M) .
- Investigate harmonization with WaterML.
- Investigate harmonization with WCS.
- Investigate harmonization between Smart Transducer Web Services (STWS)-IEE 1451 and SOS.
- Investigate using schematron to create validation rules for marine communities.
- Investigate how observations and model output could be generalized to be used in workflow compositions

## 2 Normative References

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

OGC 07-036 OpenGIS Geography Markup Language (GML) Encoding Standard, 3.2.1, 2007-08-27, [http://portal.opengeospatial.org/files/?artifact\\_id=20509](http://portal.opengeospatial.org/files/?artifact_id=20509).

OGC 07-022r1, Observations and Measurements – Part 1 - Observation schema 1.0, 2007-12-08 [http://portal.opengeospatial.org/files/?artifact\\_id=22466](http://portal.opengeospatial.org/files/?artifact_id=22466).

OGC 04-094, Web Feature Service Implementation Specification, 1.1.0, 2005-05-03, <http://www.opengeospatial.org/standards/wfs>.

OGC® 07-000, OpenGIS® Sensor Model Language (SensorML) Implementation Specification, 1.0.0, 2007-07-17, [http://portal.opengeospatial.org/files/index.php?artifact\\_id=21273&passcode=fxphjb8qrc a4gwy7g626](http://portal.opengeospatial.org/files/index.php?artifact_id=21273&passcode=fxphjb8qrc a4gwy7g626).

OGC 06-121r3 OGC Web Services Common Specification, [http://portal.opengeospatial.org/files/?artifact\\_id=20040](http://portal.opengeospatial.org/files/?artifact_id=20040).

OGC 04-095, OpenGIS Filter Encoding Implementation Specification, 1.1.0, 2005-05-03, [http://portal.opengeospatial.org/files/?artifact\\_id=8340](http://portal.opengeospatial.org/files/?artifact_id=8340).

## 3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common

Implementation Specification [OGC 06-121r3] clause 4 of Sensor Observation Service[OGC 06-009r6] and Clause 4 of Observations and Measurements – Part 1 [OGC 07-022r1].

## **4 Conventions**

### **4.1 Abbreviated terms**

API Application Programming Interface

GML Geography Markup Language

ISO International Organization for Standardization

OGC Open Geospatial Consortium

OWS OGC Web Services

OWL Web Ontology Language

O&M Observations and Measurements

MMI Marine Metadata Interoperability Project

SensorML Sensor Model Language

RDF Resource Description Framework

SAS Sensor Alert Service

SOS Sensor Observation Service

SPS Sensor Planning Service

SWE Sensor Web Enablement

TML Transducer Markup Language

UML Unified Modeling Language

XML eXtensible Markup Language

### **4.2 UML Notation**

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

## 5 Background

### 5.1 Motivation

The ocean-observing community involves scientists from academia, industry and government programs that individually deploy a wide variety of sensors. The variety introduces obstacles to creating seamless and coordinated access to data from these disparate and heterogeneous sources. Simplified data exchange would improve the way scientists observe the oceans and inform their management. Additionally, data sharing challenges become more difficult if they are to respect the need for experimental reproducibility – a hallmark of the scientific process – because different groups often represent, transport, store and distribute their data in different ways. This document describes the outcomes of an open community effort to explore the best mechanisms to make ocean data and data systems interoperable with one another.

The initiative began in 2003, when an informal community of ocean-observing programs gathered to investigate the capabilities of the WMS and WFS specifications. They achieved remarkable success, and demonstrated interoperability across institutions with web-mapping products that included in situ and satellite measurement of sea-surface temperature. Their shared website, (<http://www.openioos.org>), has been providing these real-time sea-surface temperature maps since early 2004, using readily available software. Yet, true data interoperability based on pre-existing tools remained elusive.

In 2004 NSF funded the Marine Metadata Interoperability Initiative (MMI), a community effort, whose goal is to promote agreements, standards and best practices for sharing data among the marine community. MMI develop an interoperability demonstration, using SOAP web services, Dublin Core metadata and ontologies in RDF in 2005. Some participants in this experiment were also part of the OpenIOOS community, notably those involved in a related initiative at the Southeastern Universities Research Association (SURA). In early 2006, participants agreed to combine efforts within MMI and OpenIOOS, which gave rise to OOSTethys.org. In 2007, they initiated the closely related OGC Oceans Interoperability Experiment (Oceans IE).

The timing of the OOSTethys and Oceans IE coincided with advances in the OGC Sensor Web Enablement (SWE) initiative. SWE capabilities prompted investigation between the WFS standard and the relatively new Sensor Observation Service (SOS). Extensive investigation, software development and real-world testing resulted in the set of open source SOS reference implementations and community cookbooks on OOSTethys.org. The outcomes, lessons learned, and best-practice recommendations coming from the 18-month initiative are described herein.

The OOSTethys tools were developed and tested with a distributed network of data-collecting ocean buoys. The software and services employ SensorML instance documents with information (metadata) about the platform and sensor characteristics of the various marine devices in the data network. The exchange of real-time and archived buoy measurements uses the Observation and Measurements (O&M) standard.

As the standards and experience of the different groups on publishing and accessing the

SWE standards evolve, we believe that the OOSTethys tools will continue to improve. The importance of OOSTethys and the Oceans IE could be measured by the adoption of these technologies by important ocean observing systems initiatives. SOS is being considered or adopted, at the time of writing this document, by NSF's Ocean Observing Initiative, the U.S. government Integrated Ocean Observing System, Data Integration Framework, and Europe's ESONet program.

## 5.2 User Scenario

The user scenarios driving this interoperability experiment share a common focus on exchanging and assessing processes involved in time-dependent in situ observations of marine systems. The distinguishing use cases can be extracted from the following user scenario:

A scientist has developed a decision-support tool for fisheries management. She has based her tool on research that demonstrates correlation between abyssal temperatures offshore, near shore sea-surface temperature, and various indicators of ecosystem health such as abundance and distribution of fish stocks. In particular, she knows that long-term historical records of in situ sea-surface temperature near the coast can be used to predict climatic variation in the relatively sparse measurements of offshore abyssal temperature. She needs to calibrate her decision-tool predictions using the best available long-term records and needs to base her real-time predictions of abyssal temperature in her region of interest using the best available real-time observations she can find.

She knows the data could be coming from any of several different state, federal and research institutions in the region. To find the data, she goes to a web site that allows search of a regional data-service registry created by the Gulf of Maine Ocean Data Partnership. She queries the registry for any and all in situ measurements of sea-surface temperature within 100 Km of Booth Bay in the last 20 years – she finds 200 possible sources of data. She narrows the search by selecting only those records that span more than 10 years. She'll use these data for verification of her model. She then queries the registry to refine the search to identify those measurements that are producing real-time observations. She finds 10 different data sources, and she further refines her search to find those that are producing repeated measurements at a single location, preferably a moored instrument near the bottom along the 50-meter isobath. She'll use these data for her real-time predictions.

## 5.3 Detailed Use Cases

The above user scenario includes filtering observations by different criteria, including region of interest, platform type and time. Generalizing, the uses cases can be broken down into the following set of use cases. These use cases provide functional requirements for the best practice recommendations.

1. *Find sensors/instruments in region of interest*—From a heterogeneous and distributed network of instruments & data systems, find the sensor closest to a user-defined location.
2. *Find sensors/instruments with observations in desired time range*—For a pre-

defined set of sensors/instruments, return data within user-defined time range.

3. *Get latest observation*—For a specific sensors/instruments, return the most recent observation.
4. *Get a time series of observations*—For a pre-defined set of sensors/instruments, return the time series of observations within a prescribed time range.
5. *Get some sensor info*—Return a description of the sensor/instrument used to obtain the measurement.
6. *Get quality-control info*—Return a description of the measurement process, which could include quality control procedures.

## 6 Ocean Observation and Measurement

### 6.1 Definition

An Observation is an **event** whose result is an estimate of the value of some **property** of a **feature-of-interest**, obtained using a specified **procedure**. (OGC 07-022r1). An example of an ocean observation is as follows: Buoy M, in the Monterey Bay, incorporates a SeaBird device that at 2007-01-01 12:02 PM recorded a Sea Surface Temperature (SST) of 10 degrees Celsius (C). In this example the **feature-of-interest** is the region depicted by a station located in 'Monterey Bay'. The **property** is the sea surface temperature, and the **estimate of the value** of that property is 10 deg C. The **procedure** refers to the deployment of a system encompassed by the SeaBird sensor attached to the buoy.

An ocean observing system could contain different system configurations, depending on the platform constraints and the sensing capabilities of the attached sensors. For example, a platform could be constrained by an orbit (satellite) or by a path (autonomous underwater vehicle) or by a tethered (moored buoy) or unconstrained in the horizontal plane (float). On the other hand sensors could produce not only simple point data records (fixed depth, latitude and longitude), but complex records where depth, latitude longitude could vary, such as profiles and swaths. For the purpose of this experiment we concentrated on point data records from fixed in-situ marine platforms, which the following constraints:

- Observations that produces data that do not vary in depth, latitude and longitude,
- Observations from sensors attached to fixed platforms or platforms located in the place (i.e. in-situ) where the observation is occurring.

### 6.2 Query

There are several class scenarios requiring the exchange of ocean observations. They reflect the needs of users who are performing characteristic functions like reporting, aggregating, evaluating data or assimilating data into models. A general query could be represented as follows:

Retrieve N number of records, which estimate a property P

at a geographical location L, in a given time frame T,  
collected with procedure S.

A query of this type could be seen as a set of sub queries. presents the detail of the different sub queries that we were interested in testing:

**Table 1 – Query Components**

<b>Query</b>
<b>Geographic Location</b>
lat long
lat long z
<b>Time</b>
time instance
time range
<b>Procedure</b>
1 (single procedure)
N (multiple procedures)
all available
<b>Observed Property</b>
1 (single property)
N (multiple properties)
all available
<b>Number of Records</b>
latest (1)
latest (N)

## 7 OGC Comparison result between WFS and SOS

First, an overview of the OGC technologies used is presented, then a comparison of the technologies is given.

### 7.1 Web Feature Service

Features are application objects that represent a physical entity (OGC 04-094). For example: road, station, river, coastline, etc. Features have attributes, some times called properties, such as name, shape or location. Geographic features are those features that have at least one geographic property (OGC 04-094).

The OGC Web Feature Service (OGC 04-094) provides an interface to query, as well as to perform transactions of features. Responses are encoded using the Geography Markup Language (OGC 07-036). The operations define in the interface are as follows:

- **GetCapabilities**
- **DescribeFeatureType**
- **GetFeature**
- **GetGmlObject**

- Transaction (Insert, update and delete)
- LockFeature

The first three bold interfaces are mandatory. The **getCapabilities** operation answers the capabilities of the service. For example what operations are available. The **describeFeatureType** operation answers the schema of a feature type. The **getFeature** responds to a concrete instance of a featureType and conforms to the schema answered by the **describeFeatureType**.

The operation that the Oceans IE was most interested on was **getFeature**. This operation provides the values of an instance of a feature. It is the only operation in WFS that is comparable with a **getObservation** request in an SOS service. Figure 1, presents the core components of the **getFeature** request operation of a WFS. The **getFeature** request contains one or more query elements. A query element contains a featureType, and one or more property names related to the featureType. A query also contains a filter expression, which constrains the property values using the OGC common catalog query language. Feature is a very general concept; therefore, a WFS getFeature response does not contain explicit semantics in the request and response, which could refer to the different components of an observation.

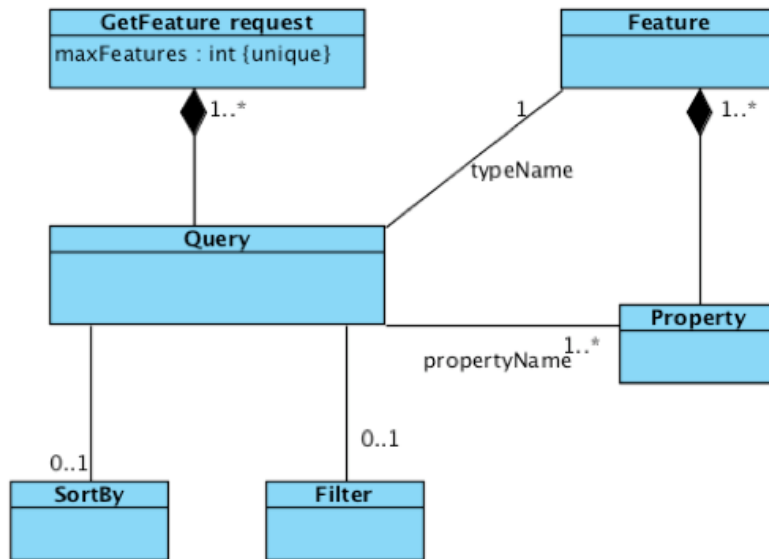


Figure 1 – Partial View of GetFeature Request

Common WFS services provide features, which are geographic “static” features, such as river lines, state boundaries polygons, etc. For example, LSU Atlas :

(<http://atlas.lsu.edu/central/displayOGCWFSfeature.htm>). And, in some cases these features have associated properties, whose values could change in time, such as observations performed in a geographic location of interest. For example, NOAA's National Digital Forecast Database ([http://www.weather.gov/xml/OGC\\_services/](http://www.weather.gov/xml/OGC_services/)), presents observations values occurring on stations, such as *temperature*, *windSpeed*, *probOfTemperatureAboveDay81*. However, presenting time series data in a WFS service is not popular, and it requires creating a complex profile.

## 7.2 Sensor Observation Service

SOS is one piece of the larger OGC Sensor Web Enablement (SWE) initiative. The goal of SWE is to enable all types of Web-enabled sensors to be accessible and, where applicable, controllable via the Web. SOS provides a broad range of interoperable capability for discovering, binding to and interrogating individual sensors, sensor platforms, or networked constellations of sensors in real-time, archived or simulated environments. Other SWE protocols include: the Sensor Planning Service (SPS), which defines an API for tasking sensor systems; and, the Sensor Alert Service (SAS) which defines interfaces for publishing and subscribing to alerts from sensors. SPS and SAS use Web Notification Service (WNS) to reformat and deliver asynchronous alerts. In addition, Transducer ML (TML) is another OGC standard to convey information about sensor systems.

The SWE operations could be categorized into core, transaction and enhanced operations.

### Core Operations (mandatory):

- GetCapabilities
- DescribeSensor
- GetObservation

### Transaction Operations (optional):

- RegisterSensor
- InsertObservation

### Enhanced Operations (optional):

- GetObservationById
- GetResult
- GetFeatureOfInterest
- GetFeatureOfInterestTime
- DescribeFeatureType
- DescribeObservationType
- DescribeResultModel

The OceansIE experiment was interested on experimenting with the core operations. For

details of the other operations the specification should be consulted (SOS).

SOS has three mandatory operations: **GetCapabilities**, **DescribeSensor**, and **GetObservation**. The **GetCapabilities** operation provides the details of the SOS service. These include: metadata about the operations, process or systems and the overview of the observation offerings.

The **DescribeSensor** operation provides detailed information about the sensor systems or processes available by the observations offerings. These include contact information, classification metadata, deployment information, capabilities, and input and output details.

The **GetObservation** operation provides access to the observed data. It contains the metadata about the observation offering, so information about the feature of interest, procedure and phenomena is provided as well.

### 7.3 Comparison of **getFeature** (WFS) with **GetObservation** (SOS)

The request and the response of the **getFeature** operation of the WFS service were compared against the request and the response of the **getObservation** operation of the SOS service.

We looked at existing WFS services (Table 2) such as NWS and SCOOP WFS and we generated new WFS and SOS services from a common data source: the Meteorological Assimilation Data Ingest System (MADIS: <http://madis.noaa.gov>), which makes value-added data available from NOAA's Earth Systems Research Laboratory. WFS was implemented using MapServer, an open-source GIS toolkit originally developed at the University of Minnesota (<http://mapserver.gis.umn.edu/>).

We set up new SOS services from different institutions, as listed in table 3. These services were created using:

- OOSTethys toolkits: open source java, python and perl libraries from the OOSTethys project. Code is available from <http://www.oostethys.org/downloads>.
- SWE Common Library: an open-source java library from the University of Alabama in Huntsville. Code is available at <http://code.google.com/p/swe-common-data-framework/>.

**Table 2 – List of investigated WFS services**

Organization	Capabilities URL
TAMU MADIS	<a href="http://sos-web.tamu.edu/sos-cgi/madis">http://sos-web.tamu.edu/sos-cgi/madis</a>
NWS NDFD	<a href="http://www.weather.gov/forecasts/xml/OGC_services/ndfdOWSserver.php">http://www.weather.gov/forecasts/xml/OGC_services/ndfdOWSserver.php</a>
MicroWFS	<a href="http://demo.asamap.com/wfs/ASA_WFS.asp">http://demo.asamap.com/wfs/ASA_WFS.asp</a>

**Table 3 – List of investigated SOS services**

Organization	Capabilities URL
TAMU MADIS	<a href="http://sos-ws.tamu.edu/tethys/madis">http://sos-ws.tamu.edu/tethys/madis</a>
MBARI SOS	<a href="http://www.mmisw.org:9600/oostethys/">http://www.mmisw.org:9600/oostethys/</a>
GOMOOS SOS	<a href="http://www.gomoos.org/cgi-bin/sos/oostethys_sos.cgi">http://www.gomoos.org/cgi-bin/sos/oostethys_sos.cgi</a>

### 7.3.1.1 Request Comparison

To perform the comparison a set of query components was formalized (Table 4). Table 4 presents the query components of and the default support of the operations of interest to respond to the query components. Default support means that the specification provides the query components without any need for extending it. It is expected that all of the services that conform to the specification should have that capability.

The main characteristic to compare was inclusiveness, or the ability to support all the query components described in . The result describes that SOS **getObservation** supports nearly all the query components given our use cases, while WFS **getFeature** only supports query by geographic location. The support for all the other query components on a WFS service depends on the extent of the definition of the feature type given by the service provider.

**Table 4 – Query decomposition and default support by WFS and SOS**

Query	getFeature() WFS 1.1	getObservation() SOS 1.0
<b>Geographic Location</b>		
lat long	X	X
lat long z	X	X
<b>Time</b>		
time instance		X
time range		X
<b>Procedure</b>		
1 (query single procedure)		X
N (query multiple procedures)		X
all available		
<b>Observed Property</b>		
1 (query single property)		X
N (query multiple properties)		X
all available		
<b>Number of Records</b>		
latest (1)		
latest (N)		

In WFS there is no concept of observation, procedure, observedProperty, time or number of records; however, a complex feature could be specified that contains all of these concepts.

In neither operation does there exist a concept for paging, or retrieving a specified number of records. For example there is no support for specifying an interval of records or asking for the latest one or latest N. These should not be confused with the capability of WFS that allows limiting the number of features. Feature, in a WFS could be an observation offering, or a procedure (e.g. buoy), but it is uncommon for a WFS to encode a time step as a feature descriptor.

### 7.3.1.2 Response Comparison

To compare the capabilities of the response with regard to our cases, we use the following two criteria:

1. Capability to express an ocean observation.
2. Capability of clients to understand responses from multiple web services (interoperability)

WFS **getFeature** response provides a feature that contains properties in GML (OGC 07-036) and that conforms to a specific profile. The 1.0.0 version of the WFS specification requires the use of GML version 2.1.2, while the 1.1.0 version of the WFS specification requires the use of GML version 3.1.1. (The latter additionally supports multiple dimensions like time and depth/elevation, and more complex geometric features.) For both versions of the WFS specification, other arbitrary encodings can be defined. A *schema* must be defined for the GML returned by WFS, so that the client can parse the returned information.

On the other hand, SOS **getObservation** response provides an observation in O&M (OGC – 07-022r1) that contains an observed Phenomena, a feature of interest, a procedure, and a result. These concepts related to an observation are given as part of the SOS standard, and there should be no need to create additional schemas or profiles. This is consistent with the GEOSS Architecture Implementation Pilot initiatives at OGC (<http://www.ogcnetwork.net/node/389>), which advises providers publishing SOS should adhere to the SOS standard as much as possible with the least amount of profiling. Additionally, the OGC OWS initiatives were not required to profile SensorML, or O&M. The ability to express an ocean observation is not a default in WFS. All the concepts must be properly designed and expressed in a custom schema that conforms to GML. Furthermore two different service providers could set up two different WFSs that could encode observations data, creating a need to develop two custom clients.

SOS has, by default, a concept of observation, and links well with related models such as the process (i.e. SensorML). SOS, which has a soft typing design, is meant to be more interoperable since a general SOS client should be able to interact with any SOS service.

## 8 Other Comparison Results

**Semantic Expression.** The expression of WFS queries is flexible, and an information model of a server can be easily expressed. In SOS, the implementer must follow best practices for using the standard under given conditions, and these do not necessarily

allow full expression of a native information model. Although to date there is limited need in the oceanographic domain for this level of semantic flexibility (especially in the context of assuring interoperable systems), more robust semantic expression of queries represents an important future need.

**Extensibility.** WFS could be extended, and indeed, is not static. SOS is meant to be used with, and possibly creating profiles with, technologies such as schematron. We envision using these profiles in schematron to create validation rules for the marine community.

**Interoperability.** For a single server running WFS or SOS, the interoperability of clients is roughly the same. The provision of the schema by the WFS server will typically provide sufficient information for the clients to automatically acquire and process information of interest, at least to roughly the same degree such computability is possible with an SOS server.

However, a case with multiple disparate data servers is somewhat different. Because the O&M schema is a widely used and flexible specification, it is more straightforward for disparate, uncoordinated collections of systems using SOS to interact, than it is for comparable systems of systems using WFS. The additional step of agreeing on a schema must be completed across the whole operational community to achieve *ad hoc* operational interoperability with WFS.

Note that in either case, interoperability may be fundamentally constrained by the need for a common semantic framework, so that clients and disparate servers use the same terms for the same concepts. As with the WFS schema, such a semantic framework must be agreed across the entire community.

**Schema Maintenance.** WFS requires the creation of an agreed schema (a 'Feature Profile') defining the server's data response, before a server can be instantiated. If a community of interest seeks standardization of their profile, such schema should be sent to the OGC standards body for approval. The development and maintenance of such schemas is a non-trivial activity, and this constrains interoperability as noted previously. On the other hand, custom development of WFS profiles may provide more flexibility to the service providers to more precisely represent their information model.

In SOS, the O&M, SensorML, and SWECommon schemas are already standards, so an initial level of schema interoperability is a given.; however, there is a need to maintain dictionaries that are used as identifiers of concepts in the services. Moreover, if communities need to agree on specific record definitions they need to define and maintain this as well, which is a slightly less burdensome task than maintaining GML schemas for WFS.

**Functional Maintenance.** The key aspect considered in this question is the difficulty of adding more capabilities, for example, additional properties, procedures, or even observation types. In WFS, any such change implies changes to the schema. As previously noted, this is a time-consuming and community-constrained activity. It may also be sensitive to the introduction of bugs and versioning issues.

SOS, on the other hand, uses a rich sensor and observation model, and is able to handle most types observations, include the marine, without any schema changes or additions. The use of soft-typing in the O&M schema enables significant flexibility to update observation responses, for example, by adding a new property offered by a sensor without requiring modification of the schema itself.

## 9 SOS Implementation Guide

This section presents implementation details for publishing an SOS Service. Since SOS, SensorML, O&M and SWE Common are recently approved OGC standards there are few comprehensive guides to their use that help in answering the best ways to utilize them. These guides are grouped by topic. Some could be considered best practices, while others will materialize in OGC change requests.

### 9.1 Modeling Systems of Systems

Figure 2 presents the conceptual model incorporating the most important components in an SOS service, as used by OOSTethys. The model complies completely with OGC standards.

#### 9.1.1 Observation

An Observation is an **event** whose **result** is an estimate of the value of some **property** of a **feature-of-interest**, obtained using a specified **procedure**.

#### 9.1.2 Feature of Interest

A Feature is a feature of any type (ISO 19109, ISO 19101). For example, the Feature body of water has properties, such as temperature and salinity. The relation is named `carrierOfCharacteristics`, which associates the class Feature with a collection of properties. Note that a property is always related to a feature. More discussion about `featureOfInterest` is provided in section 9.2.

#### 9.1.3 Procedure

SensorML provides the model to describe systems and sensors. A system is a process, and a sensor is a system. We augment the procedure class by adding concrete examples of procedures, such as `observingSystem`, `device`, `platform`, `sensor` and `detector`.

#### 9.1.4 Result

The result, based on ISO 11404, is specified by SWECommon, where `DataArrays` and `DataRecords` are defined in a more general sense. These contain Record Definitions where the field of the record is provided ( i.e. property of the `featureOfInterest`) as well as other metadata related to that record (units, datums, quality flags etc..)

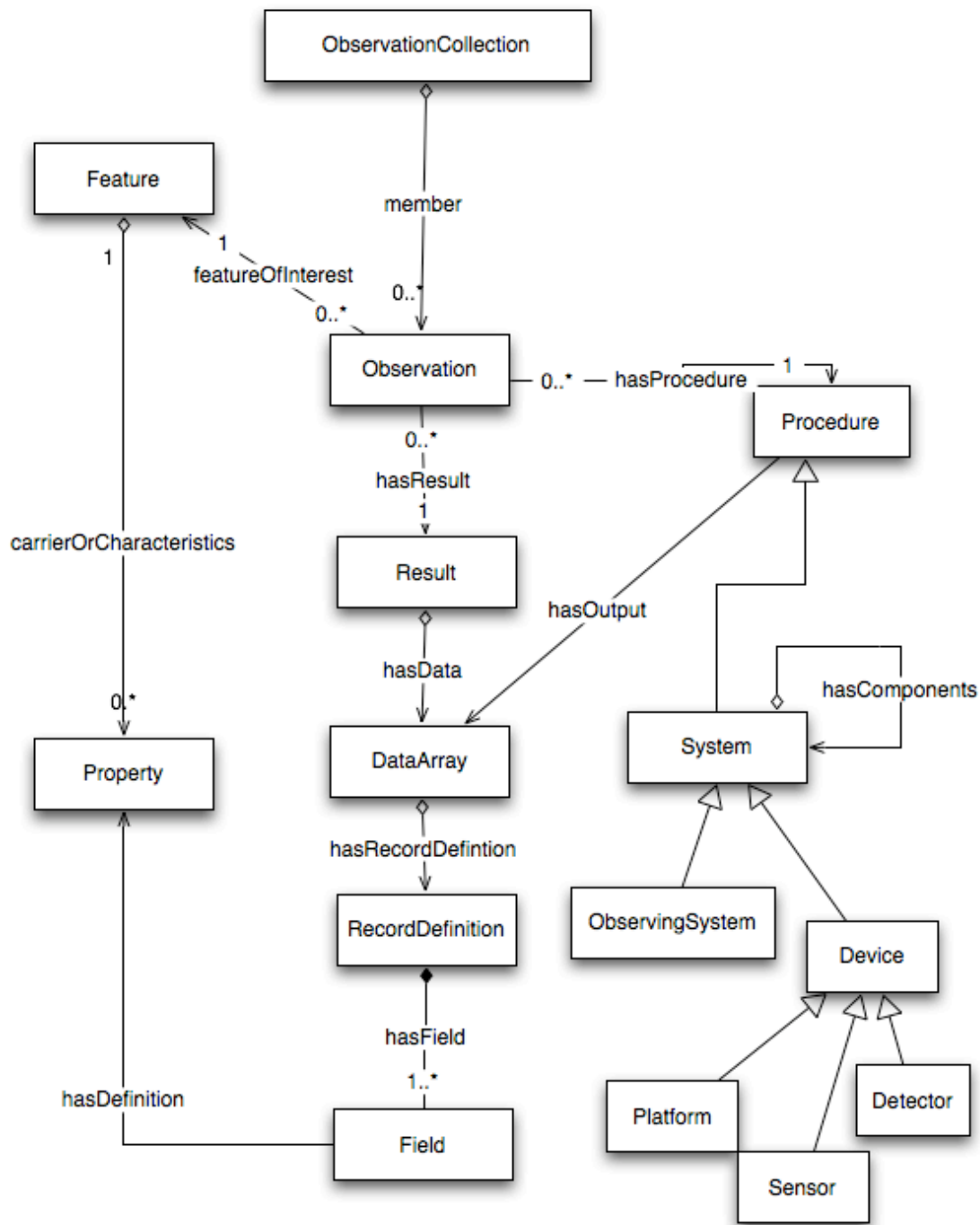


Figure 2 – Observation and System model used in OOSTethys

## 9.2 Feature of Interest

A Feature is generally defined in ISO 19109 and ISO 19101 as a thing that carries characteristics, or in other words that has properties. A property is always related to one feature.

In SOS a feature is the feature of interest of the observation. However, this definition is still too broad. The following possible explanations of feature could be depicted:

1. An earth realm: For example, ocean, river, sea surface, body of water.
  - Pros: It is similar to a feature of interest. It will always hold that the properties are related to this feature.
  - Cons: Need category of earth realms. Difficult to get the smallest encompassing realm
2. Medium: For example, air and water.
  - Pros: Easy
  - Cons: Too general. If dealing with earth observations more detailed information could be provided, as stated in 1.
3. Location
  - Pros: Information is available by the service
  - Cons: This information is already provided in the observation response, either as a point or bounding box. This could be seen as the shape of the observation, as well (see 7).
4. An event: For example, Hurricane Katrina
  - Pros: Directly expresses the event of interest.
  - Cons: Is not the feature being measured, but a process occurring in the proximity of a featureOfInterest.
5. A system: For example, platform in a fixed position in the ocean
  - Pros: Easy to understand
  - Cons: A platform doesn't have the properties being measured, so it conflicts with the model. A platform is not the feature of interest.
6. A named region: Monterey Bay, Golf of Mexico
  - Pros: Would allow direct querying for regions of interest by name, if needed.
  - Cons: This information could be inferred by a gazetteer service that could resolve locations for geographic names. Difficult to get the name for the minimum area covered.
7. Shape made by the procedure: Curve (or track) of a glider. This is similar to what O&M *part 2 (OGC 07-002r3)* discusses about the strategy suitable for estimating the observed property via the observation procedure.
  - Pros: Helps distinguishing the ultimate feature of interest with the procedure of an observation, when the procedure is confused with the feature of interest (e.g. a station).
  - Cons: Adds another level of complexity. In addition, this information could be described as a process, i.e. SensorML.

Conclusion: The Oceans IE team decided to use “earth realm” concepts as the featureOfInterest, until more precise guidance is available. Earth Realm, or any subcategory of it, is a feature of interest and so will not conflict with the O&M model.

### 9.3 Semantic Mediation and categories

There are at least two problems in environmental information systems: Semantic heterogeneity and information overload. Semantic heterogeneity occurs when there is plurality of identification of types for observation concepts (phenomena, units, properties, processes, sensors, platforms, datums, and so on). For example the phenomenon Sea Surface Temperature could have different representations, such as SST and water temperature. The problem persists even if URIs are used.

Information overload occurs when a user searches and discovers so many results that they are difficult (or impossible) to adequately filter and analyze. Categorization using Semantic Web technologies is a possible solution. Other services and systems, such as Yahoo, Ebay, and Amazon, use categorizations to help users filter the results.

In the Oceans IE we did an initial experiment to demonstrate how semantic heterogeneity and information overload could be solved, while separating the content model from the detail semantics. The soft typing of SOS allowed for service providers to add their own semantics, while achieving interoperability in the content model. The resulting content model is shown in Figure 2. The OpenIOOS web portal also created its own semantics to categorize the observations. The categories used were the main NOAA/IOOS variables (See Figure 3).

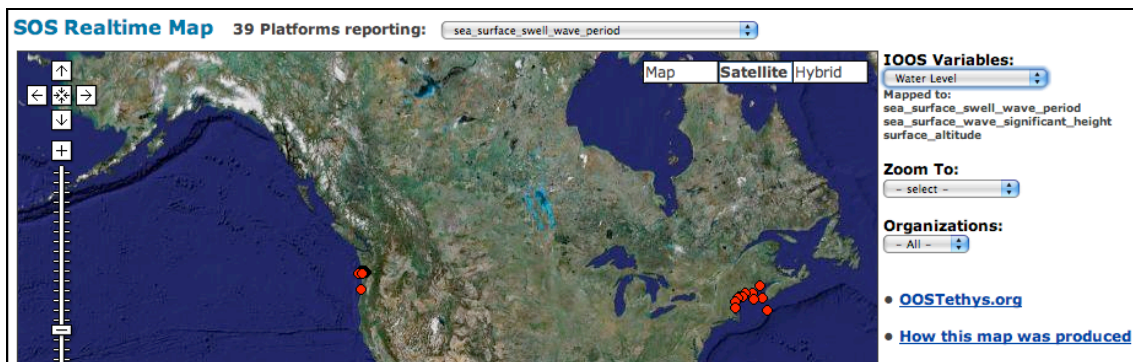


Figure 3 – Categorization of observations at openioos.org

The process could be summarized as follows:

- Creating an ontology that represents the portal categories: In this case the IOOS variables were used.
- Creating an ontology that represented the concepts used by the service providers: We suggested service providers use specific vocabularies, but this was not always followed.
- Mapping between the service providers terms and the portal categories: We used VINE (a mapping tool created by MMI) to perform such mappings
- Registering all the ontologies, including the mapping one in a registry: We used the MMI ontology registry.
- Querying of the ontologies: The OpenIOOS portal queried the registry and, using an RDF API, got the mappings of the portal terms to the services, allowing the portal to categorize the observations accordingly.

Other categories in which the portal could apply semantic mediation (and examples of each) include:

- Phenomenon types (salinity, water temperature, currents)
- Earth Realm (ocean, river, atmospheric observations)
- Platform type (research vessel, buoy, satellite)
- Quality controlled ( level 1, level 2, tests passed)
- Discipline (biology, oceanography, chemistry)
- Scientist (scientist 1, scientist 2)
- Organization (company 1, company 2, agency, project)
- Problem (coastal hazard, climate change, sea level rise)

#### 9.4 Uniform Resource Identifiers

A URI is a Uniform Resource Identifier, a concept defined by the World Wide Web Consortium. A URI is a web name for a specific resource; the resource itself may or may not be accessible via the web. A URI may be either a URL (Uniform Resource Locator), like you enter in a web browser, or a URN (Uniform Resource Name), a unique string that describes a resource.

Within SOS URIs are very important because the design of the schemas are meant to be soft typed. URIs encode the semantic information detail of the concepts being conveyed. The primary intent of a URI is to provide a unique identifier for terms. For example, it is the responsibility of the data provider to encode details about parameter names, units, quality flags, sensor types, and so on.

The common problems encountered when dealing with URIs include:

- Lack of a complete set of terminologies available in URIs.
- Lack of ability to search for the available URI encodings, if they exist.
- Lack of tools to relate terminologies from different vocabularies
- Lack of programmatic access to get URIs or to solve relations among them.

OOSTethys and the Oceans IE are working with MMI to provide a semantic infrastructure that will address the above issues. More about the MMI semantic framework and infrastructure may be found at <http://marinemetadata.org/semanticframeworkconcept>.

For the Oceans IE, all terminology used was all searchable via the MMI registry (<http://mmisw.org/registry>). The ontologies are also available via Sourceforge (at <http://mmi.svn.sourceforge.net/viewvc/mmi/mmisw/>).

In the Oceans IE service providers were not limited to vocabularies using OGC URNs, as we discovered this is not yet realistic. Among other issues, the management of large numbers of URNs for multiple organizational authorities is not yet scalable, and some details of URN definition are still being refined.

Therefore, while the next section summarizes the use of OGC URNs, a subsequent

section describes additional URI practices (adapted from MMI guidance) for other marine community users and service providers. It should be noted that service providers are not limited to use OGC URNs or MMI-recommended URIs, although there are advantages to following common practices such as those.

#### 9.4.1 How to use OGC URNs

The OGC has chosen URNs as its primary URI form for use in its specifications, has obtained a namespace for that purpose, and runs a URN resolver that can accept and resolve URNs. This section describes how to use OGC URNs. Also, if an organization wants its definitions to have the OGC namespace, then they must submit a OGC URN {ResourceSpecificString} proposal to the OGC naming authority. More information is available in the OGC URN Policy Governance document, available at the OGC web site.

The general form of a URN is urn:\$organization:string\_unique\_to\_organization The \$organization code is a code formally requested from IANA; until registered, the prefix 'x-' is prepended to the desired code. The recommended form of a URN for specifying terms in OGC is given by (OGC 05-010), as follows:

**urn:ogc:def:objectType:authority:version:code**

Considerations:

- Terms can have more than one representation (e.g., EPSG codes).
- If there is no version provided, then the version string is null and a colon may appear after another colon. For example:

urn:ogc:def:crs:EPSG::4326.

Note that OGC URN Policy and 07-092r1 are not very clear in this respect. We assume that following the OGC definition

"urn:ogc:def" ":" objectType ":" authority ":" [ version ] ":" code there must be at least six colons (":"), where:

- The version string is located between the 5th and 6th colon
- The code is located after the 6th colon
- The version number should contain no colons.

An OGC URN Scheme in Extended Backus-Nauer Form was developed for this activity, and is available here:

<http://www.oostethys.org/overview/best-practices/best-practices-ogc-urns/>

#### 9.4.2 Using user-defined dictionaries

The MMI ontology guides (<http://marinemetadata.org/apguides/ontprovidersguide>) provide a URI encoding convention for terminology developed at the MMI project. It should also help construct URIs for service providers.

MMI has recommended that controlled vocabularies (terminologies) be encoded as ontologies in some form of RDF: SKOS (Simple Knowledge Organization System) <http://www.w3.org/2004/02/skos/>, or OWL (Web Ontology Language) - <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>.

An ontology can be thought of as file, and is typically published on the web (as a file on a web site, or by an ontology repository). It is a good practice to align the topic of the ontology with the base name of the corresponding ontology file, as well as the base name of the ontology's public web (URI) representation. More information on this practice is presented under 9.4.2.3 below.

The ontology's URI construction is specified as follows:

```
MMI-URI ::= "http://" hostdomain "/" ontologiesRootDirectory "/" authority "/" [version "/"]?
           resourceType "." fileExtension
hostdomain ::= "mmisw.org"
ontologiesRootDirectory ::= "ont"
```

Example:

<http://mmisw.org/ont/mmi/200807/platforms.owl>

The parts **authority** , **version**, **resourceType** and **fileExtension** are explained in more detail below.

#### 9.4.2.1 authority

Include an authority to identify the organization that developed this version of the ontology. This helps recognize ontology owners in the URLs, helps group ontologies from the same authority, and helps distinguish similar ontologies (covering similar resources) from different authorities.

Example:

<http://mmisw.org/ont/mmi/200807/platforms.owl>

#### 9.4.2.2 version

It is optional to include a version identifier as a date for the version release, using the “YYYYMM” (year and month) pattern. We place the version number before the resourceType and fileExtension, because this puts the latter elements (which equal the ontology name and file name, see below) at the end of the path. This allows the the URI to end in .owl, as recommended below, and follows the W3C pattern.

If for any reason year and month are not sufficient to identify the file version, it is acceptable to extend the pattern through components of DD.hhmmss as needed.

Example:

<http://mmisw.org/ont/mmi/200807/platforms.owl>

#### 9.4.2.3 resourceType, ontology name, and file name

The **resourceType** should match the base file name and correspond to the file name. It should reflect the distinguishing characteristic(s) of the ontology. The resourceType can include the authority, if that aids clarity.

Choosing an appropriate resourceType helps ontology tools create a prefix of the URI namespace when opening the ontology resource. It should be a valid XML name, with no spaces or unusual punctuation. To facilitate discovery via search engines, do not use hyphens in the name.

Examples (resourceType is in bold):

**platforms.owl**, **cf\_parameters.owl**

Possible **resourcesTypes** include:

### **ISO MD\_Keywords:**

- \* Discipline
- \* Place
- \* Stratum
- \* Temporal
- \* Theme

### **OGC Object Types**

- \* axis
- \* axisDirection
- \* coordinateOperation
- \* crs
- \* cs
- \* datum
- \* dataType
- \* derivedCRSType
- \* documentType
- \* ellipsoid
- \* featureType
- \* group
- \* meaning
- \* meridian
- \* method
- \* nil
- \* parameter
- \* phenomenon
- \* pixelInCell
- \* rangeMeaning
- \* referenceSystem
- \* uom
- \* verticalDatumType

### **Other Concepts for Object Type**

- \* keyword
- \* parameter

- \* units
- \* organization
- \* platform
- \* sensor
- \* process
- \* missingflag
- \* qualityflag
- \* coordinateReference
- \* datum
- \* protocol
- \* metadataStandard
- \* featureType and featureName (e.g. gazetteer)
- \* speciesTypes and speciesNames (e.g. as used in OBIS)
- \* discipline
- \* place
- \* theme
- \* role of contact
- \* general metadata attribute

#### 9.4.2.4 fileExtension

Include the file extension, since this will help applications to understand the content of the file. For example could help search engines to access ontology file. Typing “buoy filetype:owl” in Google will query files ending with .owl with the term ‘buoy’, even though these are not currently approved MIME types.

Example:

`platforms.owl`

#### 9.4.2.5 Terms

The URL representing a term should follow the following scheme:

`http://{hostdomain}/{ontologiesRoot}/{authority}/{version}/{resourceType}#{shortName}`

All the substitution fields are the same as those described above, except for `shortName`, which is the name or code for the term in question. If *version* is omitted, the associated slash (/) is also deleted, and the URL now represents the most recently available instance of this term.

Example:

`http://mmisw.org/ont/mmi/200807/platforms#moored_buoy`

Note this scheme can be expressed as a URN:

`urn:x-namespace:ontologiesRoot:authority:version:resourceType:shortName`

The URN formulation is notional, as it depends on the existence of a URN namespace

called x-namespace (eventually 'namespace'), which has not been proposed to IETF. This form is slightly at odds with the form planned by OGC, for reasons explained above and in the MMI guidance. Since the semantic form of the URI should not be relied on to provide authoritative metadata in any case, this difference is acceptable.

### 9.5 HTTP Get Request

An HTTP request could be of various types: GET, POST, PUT etc., as explained in RFC 2616. GET passes parameters to the server in the HTTP URL (e.g. *http://marinemetadata.org:9600/oostethys/sos?VERSION=1.0.0&SERVICE=SOS&REQUEST=GetCapabilities*). POST sends a document to the server, allowing an message or result to be returned, or to provide a block of XML data, among others.

At OOSTethys we found that the GET request is very popular in the oceanographic community, because it is well-understood, easier to implement and easier to query. Sensor Observation Service, however, does not explicitly use a GET request. This section explains the GET request operations supported by OOSTethys Sensor Observation Service cookbooks. The information presented here was mostly extracted from OGC specifications and in some cases clarifications and minor comments were added. If not noted, this section complies with the OGC Web Services Common Specification 1.1.0 (OGC 06-121r3) and the Sensor Observation Service Specification 1.0 (OGC 06-009r6). If there are discrepancies between the specification and the schema, the schema is preferred.

The URL composition and the reserved characters are presented in the next two tables. The URL structure contains a host, port and path follow by a name/value pairs. The name-value pairs are called KVPs, or Keyword Value Pairs.

**Table 5 – URL Structure**

URL component	Description
http://host[:port]/path[?{name[=value]&}]	URL prefix of service operation. [ ] denotes 0 or 1 occurrence of an optional part; { } denotes 0 or more occurrences.
name=value&	One or more standard request parameter name/value pairs as defined for each operation by this International Standard. This parameter encoding is referred to as Keyword Value Pair (KVP) encoding in this document.

**Table 6 – Reserved Characters**

Character	_Reserved usage
?	Separator indicating start of query string or the KVPs.
&	Separator between the parameter (name/value) pairs in query string.
=	Separator between name and value of parameter.
,	Separator between individual values given for a parameter.

### 9.5.1 Capitalization of Keyword Value Pair names and values

Algorithms we identified for capitalizing the names and values of Keyword Value Pairs are as follows.

**Parameter names** are case insensitive. The following example shows equivalent terms for a parameter named “request”: REQUEST, request, Request, or ReQuEsT.

**Parameter values** are case sensitive and string values are [UpperCamelCase](#). These are appropriate examples for parameter values: GetObservation, DescribeSensor

### 9.5.2 Escaping special characters

[IETF 2396, section 2.4.1](#) explains that special characters should be escaped as follows: percent character "%" followed by the two hexadecimal digits. This is important because both URIs, and URNs could be sent as values for parameters, and they may include special characters

Special character	Escaped encoding
:	%3A
/	%2F
#	%23
?	%3F
=	%3D

### 9.5.3 GetCapabilities Request

The parameters rules for the getCapabilities operation are taken from table 3 and table 5 of OGC 06-121r3.

Example:

`http://host[:port]/path?request=GetCapabilities&version=1.0.0&service=sos`

**Table 7 – GetCapabilities request parameters**

Parameter Name	Definition	Data type and Values	Multiplicity and Use of the Parameter
request	Operation name	GetCapabilities	One (mandatory)
service	Service type identifier	SOS	One (mandatory)
AcceptFormats	Comma-separated prioritized sequence of zero or more response formats desired by client, with preferred formats listed first	text/xml	One (mandatory). When omitted or not supported by server, return service metadata document using MIME type "text/xml" .
Accept	Prioritized sequence of one more specification versions accepted by client, with preferred versions listed first	1.0.0 has format X.Y.Z	Zero or more (optional).

### 9.5.4 DescribeSensor Request

The attributes of the DescribeSensor request are detailed in Table 3 of the Sensor Observation Service specification 1.0 (OGC 06-009r6). The sensorId parameter specifies the sensor for which the description is to be returned. This value must match the value advertised in the xlink:href attribute of a procedure element advertised in the SOS GetCapabilities response.

**Table 8 – DescribeSensor Request Parameters**

Parameter Name	Definition	Data type and Values	Multiplicity and Use of the Parameter
request	Operation name.	describeSensor	One (mandatory)
procedure	Process or sensor system, for which the description is to be returned. It could be an observing system, platform or a simple device, such as a bin. This value must match the value advertised in the xlink:href attribute of a procedure element advertised in the SOS GetCapabilities response. In the specification, it is referred as SensorID (Table 3).	anyURI	One (mandatory)
service	Service type identifier	SOS	One (mandatory)
version	Specification version of the SOS accepted by client.	1.0.0 has format X.Y.Z	One (mandatory)
outputFormat	Specifies the desire output format.	text/xml;subtype=sensorML/1.0.1	One (mandatory)

### 9.5.5 GetObservation Request

The attributes of the GetObservation request are detailed in the Table 4 of the Sensor Observation Service specification 1.0 (OGC 06-009r6) . Here is presented a summary of the ones used by OOSTethys.

**Table 9 – GetObservation Request Parameters**

Parameter Name	Definition	Data type and Values	Multiplicity and Use of the Parameter
offering	Specifies the offering URI advertised in the GetCapabilities document. This must match the gml:name of the offering. This could also be constructed as a URL with a fragment identifier resolving to the offering gml: id. The eventTime and observedProperties are dependent on the selected offering.	anyURI	One (mandatory)
observedProperty	Comma-separated URIs, specifying the	anyURI	One (mandatory)

	phenomenon for which observations are requested.		<i>This is not mandatory in OOSTethys, If not given then all the properties is assumed.</i>
srsName	Defines the spatial reference system that should be used for any geometry that is returned in the response. This must be one of the advertised values in the offering specified in gml:srsName elements.	urn:ogc:def:crs:EPSG:6.5:4326.	One (optional). When omitted or not supported it is assume that the SRS is WGS 84 identified as urn:ogc:def:crs:EPSG:6.5:4326, which is a 2 D reference system and longitude and latitude are given in decimal degrees.
eventTime	Specifies the time period(s) for which observations are requested. This allows a client to request observations from a specific instant, multiple instances or periods of time in the past, present and future. The supported range is listed in the selected offering capabilities.	Time should conform to ISO format: YYYY-MM-DDTHH:mm:ssZ. Instance and Periods are supported. Instance is given as one time value. Periods of time (start and end) are separate by “/”. For example: 2008-04-10T10:00:00Z/2008-04-12T11:00:00Z.	One (optional). When omitted then the latest observation is returned.
BBOX	<i>This is not defined in the SOS spec. OOSTethys is using a recommended approach in: section 10.2.3 of OGC 06-121r3</i>	minlon,minlat,maxlon,maxlat,srsURI?. First 4 all in decimal degrees. srsURI is optional and could take a value of = “urn:ogc:def:crs:EPSG:6.5:4326”.	One (optional).
procedure	System for which the observations are requested	anyURI	Zero or Many (optional).
featureOfInterest	Not used at the moment. Added here as a place holder	Not applicable	Not applicable.

### 9.6 Vertical Datum Coordinate reference system

SOS, as well as other OGC specifications requires to provide a reference systems along with the coordinates when conveying geospatial information. Reference systems are encoded in the OGC services by using a code that uniquely identifies that reference system. OGC recommends to use EPSG (European Petroleum Survey Group) codes , which holds more than 11,000 codes and has a online registry available at <http://www.epsg-registry.org/>.

Describing the vertical position of marine platform and sensors is complicated by the lack of fixed and geolocatable reference systems in many marine environments. Marine platform and sensors could float on the free surface of the ocean, sit underneath a layer of water with uncertain density or be mounted on the seafloor. A common vertical

referencing problem occurs when the time-varying surface of the sea is ignore. This could happen when a floating platform is referenced to an ellipsoid or geoid, or when determining the depth of the sea for a bottom-mounted sensor.

To simplify, three base marine systems could be depicted: Sea level based systems, geoid based systems and bottom based systems. These are explained further in this section.

### 9.6.1 Sea level-based systems

Sea level-based systems refer to platforms floating on the surface of the water. The waterline of the platform may move up and down with the surface of the water. Its exact vertical location may not be known to adequate precision; however, the measurements with respect to the waterline would be well known. The EPSG code for vertical reference system referring to free surface of the water is EPSG:5113. The EPSG states this is “not recommend for use” because there is no consistent transformation from the free surface of the water to a geodetic datum. However, if the data being served is best represented as measured from a floating platform, EPSG:5113 is the best fit. As an OGC URN the combined CRS is “urn:ogc:def:crs,crs:EPSG:6.17:4326,crs:EPSG:6.17:5113”.

### 9.6.2 GeoID

The National Geodetic Survey (NGS) has adopted the following definition for geoid: “The equipotential surface of the Earth's gravity field which best fits, in a least squares sense, global mean sea level”

A common geoid vertical reference system is the North American Vertical Datum of 1988 height (NAVD88), EPSG code 5703. A combined OGC URN for WGS-84 latitude and longitude with a NAVD88 vertical height is: “urn:ogc:def:crs,crs:EPSG::4326,crs:EPSG:6.17:5703”.

Combined CRSs using geoids in other locations would replace the EPSG:6.17:5703 with the appropriate EPSG code.

### 9.6.3 Bottom-Based Systems

Bottom-based systems refer to systems on the sea floor or tethered to the sea floor. An important issue about bottom-based measurements is that you can't survey them well because of several reasons. 1) GPS signals don't penetrate the water so you can't use a GPS directly 2) depth sensors depend on the density of the water, and 3) stationing high grade GPS over a deployment long enough to get high-accuracy bottom measurement is expensive. Also, there is no EPSG code for using sea floor as a vertical reference system.

Data referenced to bottom mounted sensors may be best represented within the OGC URN system with a 2-dimensional WGS-84 CRS using OGC URN code “urn:ogc:def:crs:EPSG::4326”.

#### 9.6.4 Namespaces for Combined CRSs

OGC Namespaces for combined CRSs (07-092r1) section 7.5.1.d defines the syntax to combine CRSs. The resulting URN representations to combine WGS84 with sea level and NAVD88 are as follows:

**Combined URN for WGS84 with sea level:**

urn:ogc:def:crs,crs:EPSG:6.15:4326,crs:EPSG:6.15:5113

**Combined URN for WGS84 with NAVD88:**

urn:ogc:def:crs,crs:EPSG:6.15:4326,crs:EPSG:6.15:5703

#### 9.6.5 GetLatest

Getting the latest value of an observation is an important functionality of an observational system. For illustration, consider the display of real time status of sensors deployed in a region of interest, with their latest observations. The SOS specification doesn't provide information about how to perform such an operation, however. The questions that this section addresses is, "How do we get the value with the latest time in SOS/SWE specifications?", or "How do we get the latest time in an interval?", or "How do we obtain the latest observation(s) before a given time T?"

The following schemas were investigated, which do not reveal a best or proposed practice to get the latest observation:

- sos/1.0.0/sosGetObservation.xsd
  - (GetObservation/eventTime)
- gml/3.1.1/base/temporal.xsd
  - (RelatedTimeType/relativePosition=[Before|After|Begins|Ends|During|Equals|Contains|Overlaps|Meets|OverlappedBy|MetBy|BegunBy|EndedBy])
  - (TimeInstant, TimePeriod (with begin, end, and a duration), TimeInterval, TimePosition)
  - (TimeIndeterminateValueType=[after|before|now|unknown])
- sos/1.0.0/ogc4sos.xsd
  - (TemporalOpsType=[TM\_Before|TM\_After|...|TM\_EndedBy])
  - {note typo in line 29, 'TM\_Overalps' spelling}

It became clear in the course of discussion between OCEANS IE and SWE working group participants that the desired results are actually not about time constraints, but about ordering and limiting the result set. The desired GetLatest capabilities can be achieved with a specification that describes the ordering of results, and how many to return (e.g., 'descending time', and '1'). To get the nearest time would require introducing the concept of 'nearest', or an absolute distance from the specified time period or instant. (Similar capabilities would be needed to find the spatially nearest sample along a given dimension.)

The Oceans IE team decided on an interim practical solution by agreeing on a default behavior of the servers. If there is no time specified in the request, the response will supply the latest known observation.

## 10 Acknowledgements

The Oceans IE Team greatly appreciates the contributions of all the participants at OOSTethys. Some of them are listed as contributors in this document (section 1.2). Other contributors to this activity include: Ben Domenico (UNIDATA/UCAR), Bill Howe (NANOOS), David Coyle (USGS), Mark Wholey (ASA), Matthew Howard (TAMU), Jeremy Cothran (USC), Richard P. Signell (USGS), Paul Daisey (Image Matters LLC), and Jeff deLaBeaujardiere (NOAA/IOOS).

This activity has been supported by the National Science Foundation under award number ATM-0447031; Southeastern Universities Research Association (SURA) Monterey Bay Aquarium Research Institute and the David and Lucile Packard Foundation; Office of Naval Research, Award N00014-04-1-0721; and, by NOAA Ocean Service Award NA04NOS4730254. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.