Open Geospatial Consortium Inc.

Date: 2006-03-04

Reference number of this document: OGC 06-021r1

Version: 1.0

Category: OpenGIS® Discussion Paper

Editors: Mike Botts
Alex Robin
John Davidson
Ingo Simonis

OpenGIS® Sensor Web Enablement Architecture Document

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as 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 do

Document type: OpenGIS® Interoperability Program Report
Document subtype: Discussion Paper
Document stage: Approved
Document language: English

Contents

Page

i. Preface

Suggested additions, changes, and comments on this Discussion Paper are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

ii. Submitting organizations

The following organizations submitted this document to the Open Geospatial Consortium Inc.

Image Matters, Inc.
University of Alabama in Huntsville
University of Muenster

iii. Document contributor contact points

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

CONTACT	COMPANY	ADDRESS
Mike Botts	University of Alabama in Huntsville	ESSC/NSSTC Huntsville, AL 35899
Alex Robin	University of Alabama in Huntsville	ESSC/NSSTC Huntsville, AL 35899
John Davidson	Image Matters	201 Loudoun St, SW Leesburg, VA 20175
Ingo Simonis	University of Muenster	Robert-Koch-Str. 26-28 48149 Muenster Germany
Simon Cox	CSIRO Exploration and Mining	ARRC PO Box 1130 Bentley WA 6102 Australia

iv. Revision history

Date	Release	Editor	Primary clauses modified	Description
2006-01-20	R1	meb	throughout	Initial draft
2006-02-13	R1	meb, ar	throughout	completed draft

v. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document.

vi. Future work

Improvements in this document are desirable to account for status changes in the individual specifications and to outline further use cases within Appendix A.

Foreword

There are no annexes to this document. This is an Informative document that describes the framework surrounding sensor web/network specification described in other OGC documents. Therefore, no Sections nor Annexes in this document are normative.

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

Introduction

In much the same way that HTML and HTTP standards enabled the exchange of any type of information on the Web, the Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) initiative is focused on developing standards to enable the discovery, exchange, and processing of sensor observations, as well as the tasking of sensor systems. The functionality that OCG has targeted within a sensor web includes:

- Discovery of sensor systems, observations, and observation processes that meet our immediate needs
- Determination of a sensor's capabilities and quality of measurements
- Access to sensor parameters that automatically allow software to process and geolocate observations
- Retrieval of real-time or time-series observations and coverages in standard encodings
- Tasking of sensors to acquire observations of interest
- Subscription to and publishing of alerts to be issued by sensors or sensor services based upon certain criteria

Within the SWE initiative, the enablement of such sensor webs and networks is being pursued through the establishment of several encodings for describing sensors and sensor observations, and through several standard interface definitions for web services. Sensor Web Enablement standards that have been built and prototyped by members of the OGC include the following pending OpenGIS Specifications:

1. **Observations & Measurements (O&M) -** The general models and XML encodings for observations and measurements made using sensors. [OGC 05-087]

2. **Sensor Model Language (SensorML) –** standard models and XML Schema for describing the processes within sensor and observation processing systems; provides information needed for discovery, georeferencing, and processing of observations, as well as tasking sensors and simulations. [OGC 05-086]

3. **Transducer Model Language (TML) –** conceptual model and XML Schema for describing transducers and supporting real-time streaming of data to and from sensor systems. [OGC 05-085]

4. **Sensor Observation Service (SOS) –** An open interface for a service by which a client can obtain observations and sensor and platform descriptions from one or more sensors. [OGC 05-088]

5. **Sensor Planning Service (SPS) –** An open interface for a service by which a client can 1) determine the feasibility of collecting data from one or more sensors

or models and 2) submit collection requests to these sensors and configurable processes. [OGC 05-089]

6. **Sensor Alert Service (SAS)** – An open interface for a web service for publishing of and subscribing to deliverable alerts from sensor or simulation systems. [OGC 05-098]

7. **Web Notification Service (WNS)** – An open interface for a service by which a client may conduct asynchronous dialogues (message interchanges) with one or more other services. [OGC 05-114]

The sensor web standards infrastructure defined by these specifications constitutes a revolution in the discovery, assessment and control of live data sources and archived sensor data. The goal of this document is to discuss design and operational concepts for the SWE Architecture.

OpenGIS® Sensor Web Enablement Architecture Document

1 Scope

The aim of this document is to provide a overview description of the general architecture that applies to the Sensor Web Enablement (SWE). While this document provides a synopsis of the relevant encodings and web services, it does not contain interface descriptions of the components. These are provided within other documents, as specified below.

OGC 06-010, *OpenGIS® Transducer Model Language (TransducerML)*

OGC 05-086r2, *OpenGIS® Sensor Model Language (SensorML)*

OGC 05-087r2, *OpenGIS® Observations and Measurements (O&M)*

OGC 06-009, *OpenGIS® Sensor Observation Service (SOS)*

OGC 05-089r2, *OpenGIS® Sensor Planning Service (SPS)*

OGC 05-098, OpenGIS® *Sensor Alert Service (SAS)*

OGC 05-114, *OpenGIS® Web Notification Service (SPS)*

2 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of 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 19105:2000, *Geographic information — Conformance and Testing*

OGC 05-008, *OpenGIS® Web Services Common Specification*

This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

3 Terms and definitions

For the purposes of this document, the definitions specified in Clause 4 of the OWS Common Implementation Specification [OGC 05-008] and shall apply. In addition, the following terms and definitions apply.

4.1
application schema
conceptual schema for data required by one or more applications

[ISO 19101]

4.2
GML application schema
application schema written in XML Schema according to the rules specified in ISO 19136

[ISO 19136]

4.3
association
semantic relationship between two or more classifiers that specifies connections among their instances

[ISO 19501-1]

A binary association is an association among exactly two classifiers (including the possibility of an association from a classifier to itself).

4.4
attribute <UML>
feature within a classifier that describes a range of values that instances of the classifier may hold

An attribute is semantically equivalent to a composition association; however, the intent and usage is normally different.

4.5
attribute <XML>
name-value pair contained in an **element**

4.6
child element <XML>
immediate descendant **element** of an **element**

4.7
codespace
rule or authority for a code, name, term or category

EXAMPLE Examples of codespaces include dictionaries, authorities, codelists, patterns, etc.

4.8
coordinate reference system
coordinate system that is related to the real world by a datum [ISO 19111]

4.9
coverage
feature that acts as a function to return values from its range for any direct position within its spatiotemporal domain

[ISO 19123]

4.10
data type
specification of a value domain with operations allowed on values in this domain

[ISO/TS 19103]

EXAMPLE Integer, Real, Boolean, String, Date (conversion of a data into a series of codes).

Data types include primitive predefined types and user-definable types. All instances of a data types lack identity.

4.11
determinand
parameter or a characteristic of a phenomenon subject to **observation.** Synonym for **observable.**

4.12
domain
well-defined **set**

[ISO/TS 19103]

1 A mathematical **function** may be defined on this set, i.e. in a function f:A→B A is the domain of the function f.

2 A domain as in domain of discourse refers to a subject or area of interest.

4.13
element <XML>
basic information item of an XML document containing **child elements**, **attributes** and character data

From the XML Information Set: “Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, sometimes called its ‘generic identifier’ (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value.”

4.14
feature
abstraction of real world phenomena

[ISO 19101]

A feature may occur as a type or an instance. Feature type or feature instance should be used when only one is meant.

4.15
feature association
relationship that links instances of one feature type with instances of the same or different feature type

[ISO 19110]

4.16
grid
network composed of two or more sets of **curves** in which the members of each set intersect the members of the other sets in an algorithmic way

[ISO 19123]

The curves partition a space into grid cells.

4.17
measure (noun)
value described using a numeric amount with a scale or using a scalar reference system

[ISO/TS 19103]

When used as a noun, measure is a synonym for physical quantity.

4.18
measurement (noun)
an observation whose result is a measure

4.19
measurand
physical parameter or a characteristic of a phenomenon subject to a measurement

4.20
namespace <XML>
collection of names, identified by a URI reference, which are used in XML documents as element names and attribute names [W3C XML Namespaces]

4.21
observable (noun)
parameter or a characteristic of a phenomenon subject to **observation**

4.22
observation (noun)
an act of observing a property or phenomenon, with the goal of producing an estimate of the value of the property

4.23
property <General Feature Model>
characteristic of a feature type, including attribute, association role, defined behaviour, feature association, specialization and generalization relationship, constraints

[ISO 19109]

4.24
property <GML>
a **child element** of a GML object

It corresponds to feature attribute and feature association role in ISO 19109. If a GML property of a feature has an xlink:href attribute that references a feature, the property represents a feature association role.

4.25
range
set of all values a function f can take as its arguments vary over its domain

4.26
result
an estimate of the value of some property

4.27
scale
a particular way of assigning numbers or symbols to measure something is called a *scale* of measurement [SAR1995].

4.28
schema
formal description of a model

[ISO 19101]

In general, a schema is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the relationship between the attributes and elements of an XML object (for example, a document or a portion of a document)

4.29
schema <XML Schema>
collection of schema components within the same target **namespace**

EXAMPLE Schema components of W3C XML Schema are types, elements, attributes, groups, etc.

4.30
schema document <XML Schema>
XML document containing schema component definitions and declarations

The W3C XML Schema provides an XML interchange format for schema information. A single schema document provides descriptions of components associated with a single XML namespace, but several documents may describe components in the same schema, i.e. the same target namespace.

4.31
semantic type
category of objects that share some common characteristics and are thus given an identifying type name in a particular domain of discourse

4.32
sequence
finite, ordered collection of related items (objects or values) that may be repeated

[ISO 19107]

4.33
set
unordered collection of related items (**objects** or values) with no repetition

[ISO 19107]

4.34
tag <XML>
markup in an XML document delimiting the content of an **element**

EXAMPLE <Road>

A tag with no forward slash (e.g. <Road>) is called a start-tag (also opening tag), and one with a forward slash (e.g. </Road> is called an end-tag (also closing tag).

4.35
tuple
ordered list of values

4.36
UML application schema
application schema written in UML according to ISO 19109

4.37
Uniform Resource Identifier (URI)
unique identifier for a resource, structured in conformance with IETF RFC 2396

The general syntax is <scheme>::<scheme-specific-part>. The hierarchical syntax with a namespace is <scheme>://<authority><path>?<query> - see [RFC 2396].

4.38
Value
member of the value-space of a datatype. A value may use one of a variety of scales including nominal, ordinal, ratio and interval, spatial and temporal. Primitive datatypes may be combined to form aggregate datatypes with aggregate values, including vectors, tensors and images [ISO11404].

4 Conventions

4.1 Abbreviated terms

API	Application Program Interface
COTS	Commercial Off The Shelf
O&M	Observations and Measurements
OWS3	Open Web Services Initiative 3.0 (March – November 2005)
SAS	Sensor Alert Service
SensorML	Sensor Model Language
SOS	Sensor Observation Service
SPS	Sensor Planning Service
SWE	Sensor Web Enablement
TML	Transducer Model Language
WNS	Web Notification Service

4.2 UML notation

Most diagrams that appear in this specification are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 05-008].

5 Sensor Web Enablement Architecture Overview

The Sensor Web Enablement (SWE) architecture was designed to enable the creation of web-accessible sensor assets through common interfaces and encodings. Sensor assets may include the sensors themselves, observation archives, simulations, and observation processing algorithms. The role of SWE is depicted in Figure 1. SWE not only enables interoperability among disparate networks of sensors and among disparate models and simulations, but it also enables increased interoperability between sensors and models, and between these and the decision support tools where the final application of observations occurs.

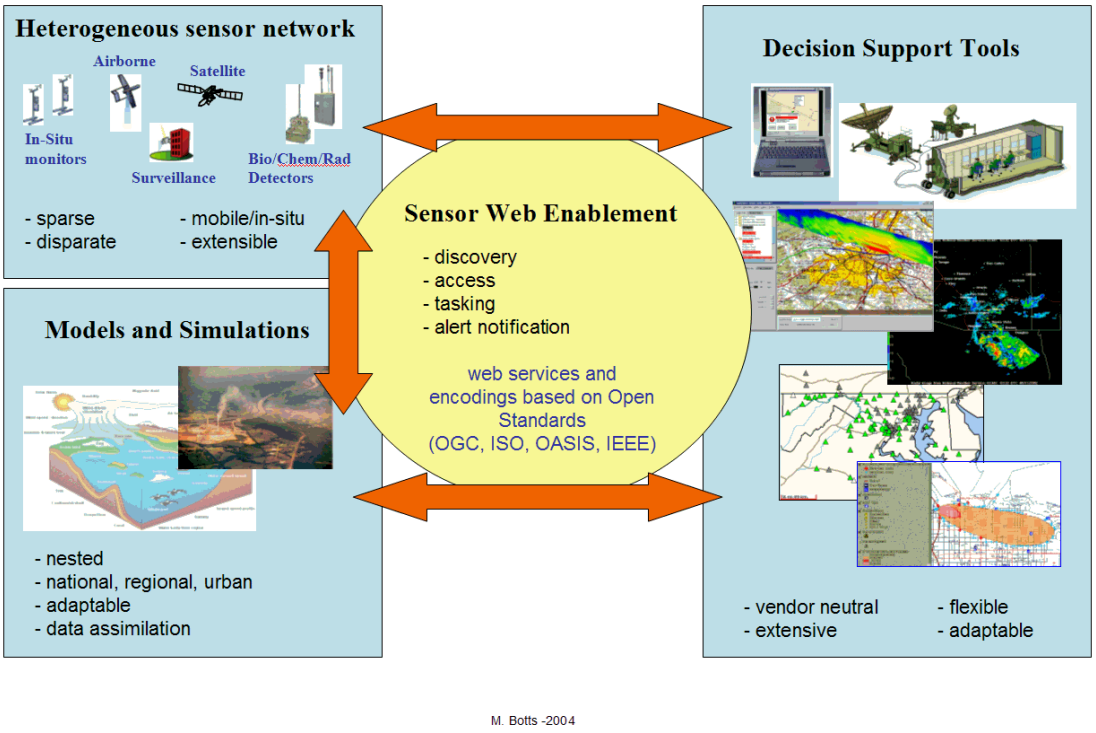


Figure 1. Role of SWE

The role of the OGC Sensor Web Enablement framework is to provide interoperability among disparate sensors and models, as well as to serve as an interoperable bridge between sensors, model and simulations, and decision support tools.

5.1 Motivation

It has long been recognized that the current state of sensor networks is that they are developed around different communities of sensor types and user types, with each

community typically relying on its own stovepipe system for discovery, accessing observations, receiving alerts, and tasking sensor systems and models. Even within fairly coherent communities, each type of sensor tends to be accompanied by its on metadata semantics, its own data formats, and its own software.

Within such stovepipe systems, the ability to discover and utilize a new sensor asset is typically hindered by incompatible encodings and services. Additionally, readily available information regarding the sensor system, the observation encodings, processing, and supporting services is typically lacking, scattered, or incomplete. Within these systems, adding support for a new sensor asset to an existing decision support tool or processing operation takes at best several days, and at worst many months or years, accompanied by high expense.

Perhaps most significantly, the conventional systems are provider-oriented. The primary classification is the specific sensor type, and the user of sensor information is expected to judge its appropriate application. To cognoscenti the name of the sensor is indeed an effective proxy for knowledge of the capabilities of the system. However, for more casual users, classification of a sensor service in terms of more semantically explicit classifiers (e.g. target feature, observed property) supports a more suitable discovery and request interface.

As described in the Introduction, the Sensor Web Enablement (SWE) framework has been designed to enable solutions that meet the following desires:

- **Discovery of sensors, observations, and processes** – we wish to enable one to easily discover all sensor assets (sensor systems, simulations, and data processes) that are available for meeting that individuals needs in a timely fashion; this is particularly important, for example, during rescue or mitigation operations following an unexpected disaster or attack
- **Determination of a sensor's capabilities and an observation's reliability** – we wish to provide the ability to readily assess the capabilities of a sensor or simulation system, as well as provide sufficient lineage of an observation to determine its reliability for decision support
- **Access to parameters and processes that allow on-demand processing of observations** – we wish to provide the means to sufficiently be able to support on-demand geolocation and processing of sensor observations by generic software, without the need for a priori knowledge of the sensor system
- **Retrieval of real-time or time-series observations in standard encodings** – we wish to be able to access and immediately utilize observations from newly discovered sensors within decision support tools, models, and simulations without needing to develop sensor-specific readers
- **Tasking of sensors and simulators to acquire observations of interest** – we wish to be able to task a sensor or simulation system, and to provide my collection requirements, using a common interface; this interface needs to be able to support

tasking as simple as controlling a web cam, as well as something as sophisticated as a military surveillance operation

- **Subscription to and publishing of alerts based on sensor or simulation observations** - we wish to provide a means by which a sensor system or simulation can publish possible alerts to be issued by sensors or sensor services based upon certain criteria, and allow one to subscribe to and receive these alerts when criteria are met; such criteria could be a simple as a measured value exceeding a certain threshold or as complex as pattern recognition within a single or multiple observations

5.2 Approach

Within the SWE initiative, the enablement of such sensor webs is being pursued through the establishment of three encodings for describing sensors and sensor observations, and through four standard interface definitions for web services:

Observations and Measurements Schema (O&M) – standard models and XML Schema for encoding observations and measurements from a sensor, both archived and real-time

Sensor Model Language (SensorML) – standard models and XML Schema for describing sensors systems and processes; provides information needed for discovery of sensors, location of sensor observations, processing of low-level sensor observations, and listing taskable properties

Transducer Model Language (TransducerML) – the conceptual model and XML Schema for describing transducers and supporting real-time streaming of data to and from sensor systems

Sensor Observations Service (SOS) – standard web service interface for requesting, filtering, and retrieving observations and sensor system information

Sensor Planning Service (SPS) – standard web service interface for requesting user-driven acquisitions and observations

Sensor Alert Service (SAS) – standard web service interface for publishing and subscribing to alerts from sensors

Web Notification Services (WNS) – standard web service interface for asynchronous delivery of messages or alerts from SAS and SPS web services and other elements of service workflows.

5.3 High level concepts

The goal of SWE is to make all types of network-resident sensors, instruments, imaging devices and repositories of sensor data, discoverable, accessible and, where applicable,

controllable across scalable networks. That is, the goal is to enable the creation of Web-based sensor networks.

It is becoming increasingly practical to fit sensors of virtually any type with wired or wireless connections that enable remote access to the devices' control inputs and data outputs and their identification and location information. Through a variety of location technologies such as GPS, IMU, Cell-ID, and Cell-ID with triangulation, mobile sensing devices can be made capable of reporting their geographic location along with other sensor-collected data.

If the connection can be layered with Internet and Web protocols, eXtensible Markup Language (XML) can be used to publish formal descriptions of the sensor's capabilities, location, and interfaces. Then Web clients, servers, brokers and devices can make their existance known, enabling automated Web-based discovery of sensors and evaluation of their characteristics based on their published descriptions. Information provided in the XML descriptions about a sensor's control interface enables automated communication with the sensor system to determine, for example, its state and location, to issue commands to the sensor or its platform and to access its stored or real-time data. This service-oriented approach to sensor and data description also provides a very efficient way to generate, publish and discover comprehensive standard metadata for data produced by sensors and processes.

Figure 1 depicts two key concepts of SWE networks enabled with open, standardized Web service interfaces and data encodings:

- Multi-level integration
- Sensor-observation value chains

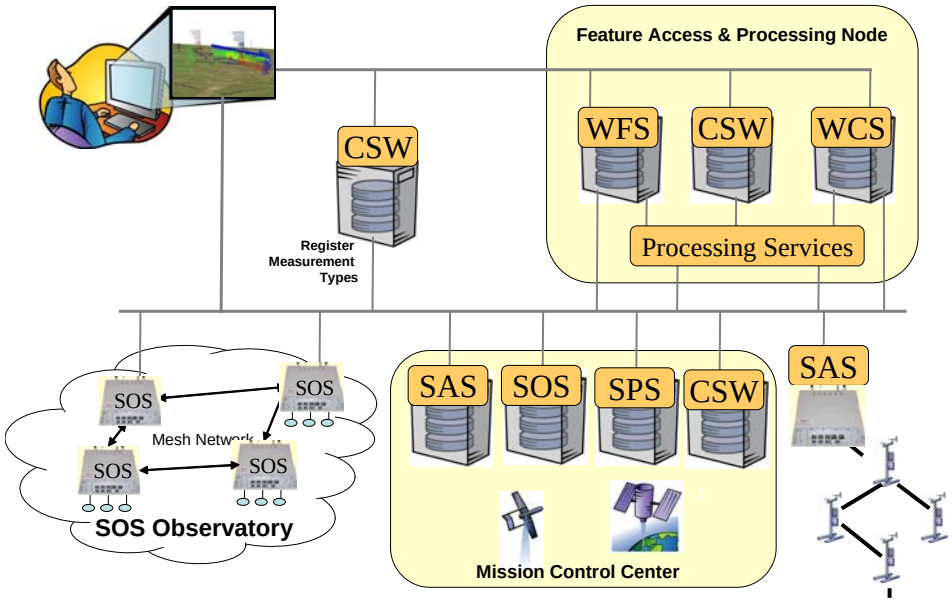


Figure 1. SWE Concept

Multi-level integration.

By multi-level integration we mean the ability for multiple information communities, organizations, systems and sensors to access and share resources (data, devices, processes, systems and services) over common or interconnected networks. In these "ecosystems" comprised of devices, multi-mode networks, databases, processors and applications, connectivity and data-sharing can occur dynamically at varying intervals, volumes and rates. A single SWE ecosystem may be comprised of, for example, a wireless mesh of thousands of networked peer-to-peer sensor devices, a single constellation of fixed in-situ sensors connected to a single network-enabled node and a large mission-control center for managing the planning and deployment of mobile sensors (remote and in-situ) with a ground station for collection and storage of the raw (pre-processed) data they produce. Each of these sensor system and processing nodes may be owned and operated by different organizations and for different purposes such as chemical-biological-nuclear-radiological-explosion (CBRNE) detection, perimeter intrusion detection, environmental monitoring, scientific research, etc). The SWE concept for multi-level integration facilitates scalable, maintainable and extensible access to resources.

Sensor-observation value chains.

A value chain is variously defined as[1]:

- The sequential set of primary and support activities that an enterprise performs to turn inputs into value-added outputs for its external customers or to add indirect value by supporting other enterprise operations
- The set of linked, value-creating activities, ranging from securing basic raw materials and energy to the ultimate delivery of products and services
- The set of activities required to design, procure, produce, market, distribute, and service a product or service
- A linked set of activities within a supply chain that actively add value to the end product
- Identifying the primary activities that create value for customers and the related support activities.

In the context of SWE and as depicted in Figure 1, the sensor-observation value chain is about the lifecycle of sensor-produced observations, from raw unprocessed data granules to information products and services delivered to applications and consumers. In between these ends of the value chain, orchestration of linked services to process the raw data into representations that are suitable for exploitation by different organizations and for different purposes occurs. Depending on the requirements, sensor data may be delivered to the user in its most elemental unprocessed form, processed into an observation object complete with the metadata and processes used to estimate a value describing a phenomenon, or processed further, for example, into single-valued geographic feature or multi-valued coverage representations. In Figure 1, value-added processing may be provisioned onboard sensors, at sensor proxy nodes, at ground stations, within mission control centers, at storage and processing nodes of the network, or on users' desktops. Standard service interfaces and data encodings, such as those defined by SWE, are the key to large-scale and extensible deployments of sensor-observation value chains.

5.4 Example Scenarios

As an example of a use-case scenario in which a SWE-enabled sensor web capabilities might support, consider the need for remedial action following a massive explosion and the subsequent widespread burning of toxic materials. A small sampling of the interactions with web-enabled sensor components could include:

1. an emergency management team (e.g. FEMA) uses sensor registry to discover that NASA has an airborne sensor capable of providing thermal imaging through smoke and clouds (note: there was an actual need to discover such a sensor following the WTC attacks, but the capabilities of and access to the sensor were not discovered until many days after the real need passed) … team places a request for acquisition through SPS and within the hour receives notification through WNS that task is approved and scheduled

[1] From Google searches for "value chain"

2. wind profilers in surrounding area are discovered through sensor registries… measurements obtained (through SOS), geolocated (using SensorML), and applied to simulations of aerosol plume transport … results predict the downwind dispersion location of toxic cloud over the next 6-12 hours

3. biochemical "sniffers" are discovered that are available through secure connections as part of the homeland security initiative .... sensor capabilities are queried and it is determined (using SensorML) that these sensors are capable of detecting the chemical species in question and have high enough sensitivities to detect probable concentrations … team subscribes to alerts when concentrations above a specific value are detected (notice to be sent by email/SMS (Short Message Service) to FEMA personnel and by URL to plume simulation service for automatic updating and refinement of its model runs)

4. through a sensor registry, mobile sensors for monitoring toxic aerosols are discovered to be available through a regional commercial emergency response team … monitor deployment requested and authorized through SPS … FEMA subscribes to alerts from these sensors when certain concentration thresholds are exceeded … alerts sent automatically (by WAS) from field sensors via wireless network along with locations and measurements … containment and medical teams sent to appropriate areas to tend to the local situation

5. in meantime, thermal imager observations taken … notifications sent (via WNS) to fire and hazards teams … unprocessed imagery (streaming as TML data) is obtained through SOS, as are process descriptions (in SensorML) that allow on-demand geolocation and processing of thermal data … location of main hot-spots and source of main toxic located … fire containment is focused on these areas

6. sensors onboard emergence response vehicles monitor not only the location of these vehicles at all times, but some also provide mobile measurement of airborne toxin concentrations; real-time video streams (TML) from sensors onboard UAVs and other airborne platforms are provided through SOS services along with SensorML processes for geolocation of video, allowing real-time hazard assessment and rescue monitoring

7. all of these data and services are available to one or more SWE-enabled decision support tools that are capable of processing and fusing this information without *a priori* knowledge of any of the sensor systems involved

This is, of course, only a single possible scenario of the application of SWE supported assets. Additional scenarios have been created for autonomous sensor webs where sensor systems are capable of subscribing to alerts from other sensors and modifying their own sensing behavior based on these alerts.

6 SWE Architecture Components

The SWE components listed in Section 5.2 are defined in detail within separate documents. The following provides an informative description and status of each component within the SWE Architecture.

6.1 Information Model

The SWE Information Model is comprised of conceptual components that individually address specific functional and system aspects of SWE systems. One way to understand the SWE Information Model is by its high-level logical components and their relationships. These conceptual elements are (in "raw" to "processed" order):

- **Transducer**: A physical or virtual entity capable of translating between physical phenomenon and transducer data. Transducers that translate phenomenon to data are typically referred to as receivers or sensors, and transducers that translate data to phenomenon are typically referred to as transmitters or actuators. Transducers are the interface between the digital realm and the physical realm or real world. Anytime a computing machine is processing data, its input is received through a transducer and it output it through a transducer. [05-085]

- **Process**: Takes one or more inputs and, based on parameters and methodologies, generates one or more outputs. [05-086r2]

- **System**: A composite model of a group or array of components, which can include detectors, actuators or sub-systems. A system relates a Process Chain to the real world and provides additional definitions regarding relative positions of its components and communication interfaces. [05-086r2]

- **Observation**: An act of observing a property or phenomenon, with the goal of producing an estimate of the value of the property. A kind of event and thus is associated with a discrete time. An observable is a parameter or a characteristic of a phenomenon subject to observation, [05-087r2]

In Figure 2, we show these logical components as layers that encapsulate data and behaviors into increasingly high-order representations, from the transducer at the physical/digital boundary to application clients that consume observations and other system data to produce and deliver processed and packaged information to end users.

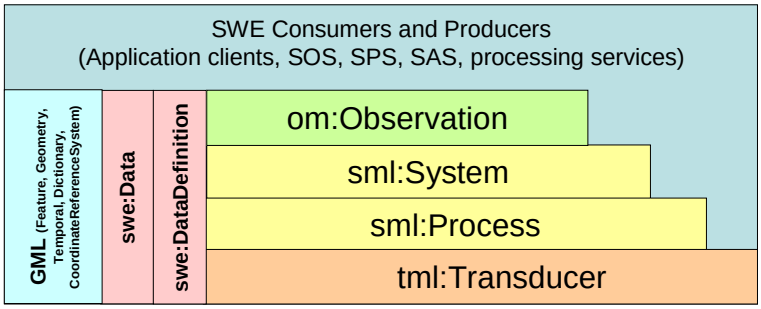


Figure 2. Logical Layers of the SWE Information Model

Figure 2 also introduces the SWE data stack, the semantic and structural description of the data components (e.g., phenomenon, units of measure, position, data type, etc), encoding schemes and values used to represent data inputs and outputs across the logical layers of the SWE system.

The logical layer model relates the SWE information model components to each other and to the SWE value chain. It should be interpreted not as a physical model but rather a conceptual one. It represents a model for interfacing applications to key information elements of the SWE architecture. As illustrated, applications (user-facing clients or other processing services) may interact with all layers of the SWE Information Model, other OGC-defined objects and external systems and representations as needed. In this viewpoint, applications should restrict themselves to the SWE information component layers whenever possible. This restriction aids in the portability of applications and components across multiple SWE-based systems. In addition, higher-level components of the information model should be used whenever possible or practicable. The higher-level components of the model (e.g., Observation) typically hide a great number of details and complexities inherent in large-scale sensor systems found in the lower level components. Following this guideline frees applications (and application developers) to focus on their objectives rather than building infrastructure to handle the details themselves.

The layers of the model are not exclusive and all or combinations of layers may be used in processing and analysis. If, therefore, the details of the transducer response model or the process and system characteristics must be determined by an application, the SWE Information Model provides a common vocabulary and representation for handling this information.

Note also in Figure 2 that the “swe:DataDefinition” and “swe:Data” components (defined in the “SWE Common” schema) specify common ways in which data values are described, encoded and used as process inputs, parameters and observation results. Note too that these SWE components depend on and extend core elements defined by the OGC’s GML conceptual model and schema. All components of the SWE Information Model, including the top-most application layer, use these common components.

6.1.1 Observations and Measurements (O&M) [OGC 05-087r2]

Through the O&M specification, the SWE framework provides a standard model for representing and exchanging observation results. O&M is primarily a conceptual model describing the relationship between different aspects of the data-capture process, specially separating the concerns of a user-centric viewpoint. The modeling approach that follows ISO 19103, ISO 19118 and ISO 19136 also lends itself to direct XML serialization as a GML application schema.

O&M provides standard constructs for accessing and exchanging observations, alleviating the need to support a wide range of sensor-specific and community-specific data formats. Particularly with advancements made during the OWS3 project, the O&M *Observation* provides a standard that combines the flexibility and extensibility provided by XML with an efficient means to package large amounts of data as ASCII or binary blocks.

An observation is a different but complementary representation of spatio-temporal phenomena than the other more traditional forms: *feature* and *coverage*. A feature can generally be considered a single, constant-valued object. A coverage has multiple property values in the spatio-temporal domain. An observation is an event with metadata and procedures for estimating the value of phenomena. Each of these representations has specific conceptual utility for certain data collection, handling, geoprocessing, analysis and dissemination activities. At times it is important to clearly distinguish and use one or more of these forms to provide different representation of what is essentially the same thing.

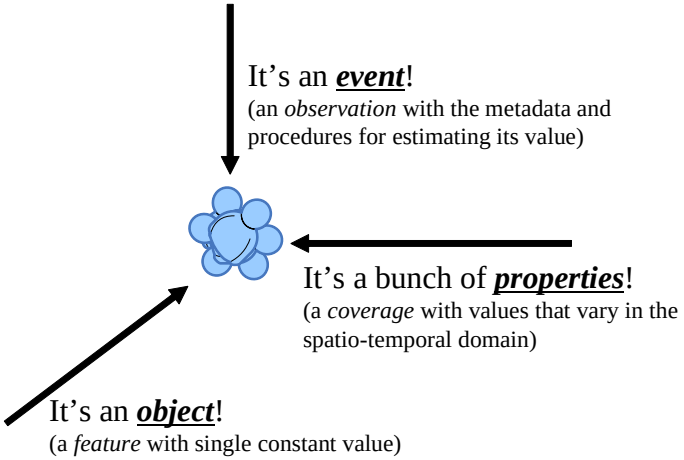


Figure 3. Feature, Coverage and Observation Representations.

As defined within the O&M specification, an *Observation* is an event with a *result* that has a value describing some phenomenon. The observation is modeled as a Feature within the context of the ISO/OGC Feature Model. An observation feature binds the result to the feature of interest, upon which it was made. An observation uses a procedure to determine the value, which may involve a sensor or observer, analytical procedure, simulation or other numerical process. This procedure would typically be described as a process within SensorML. The observation pattern and feature is primarily useful for capturing metadata associated with data capture.

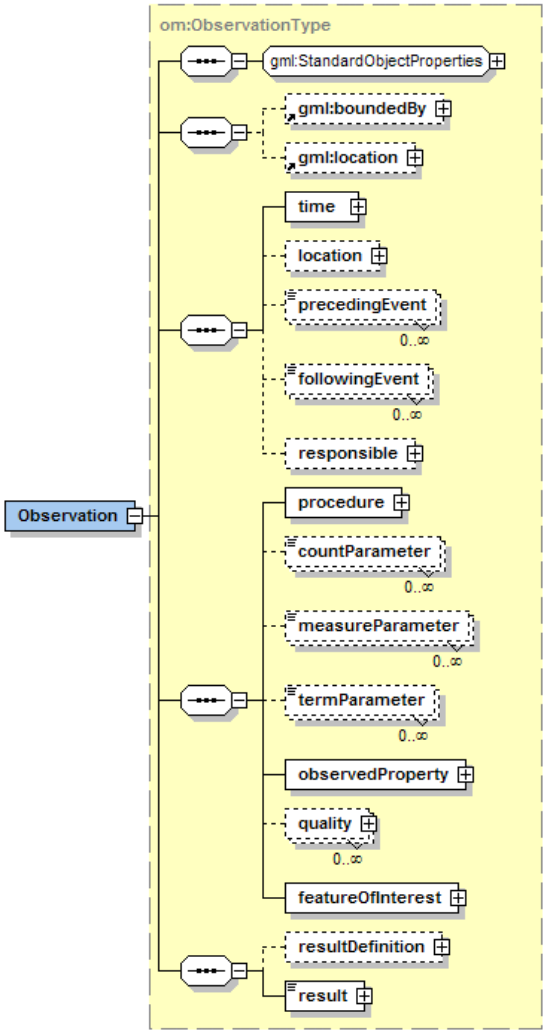


Figure 4. Base properties of the O&M *Observation* describing the time, location, feature of interest, procedure, observed property, quality, and result of an observation event.

The O&M specification allows for extension of the *Observation* object to support various styles of providing observation result values. This is an area of current research to define styles that adequately and efficiently support both simple and complex results, as well as perhaps legacy formats and out-of-band data. These observation models typical differ only in the encoding of the *resultDefinition* and *result* properties. All other properties of the *Observation* model remain common for all derived observation types.

During the OWS3 project, a *CommonObservation* model was defined in order to support a wide variety of sensor data with a common model. This observation type utilized the *DataDefinition* and basic data types defined in the SWE Common namespace and used throughout the SWE architecture. The SWE Common *DataDefinition* object separates the descriptions of the data components from the description of the encoding of the values. The *CommonObservation* model was utilized throughout the OWS3 test bed and was shown to efficiently support both ASCII and binary encoding of large blocks of data. The model supports out-of-band streaming of data (e.g. TML multiplex data) and simple "mime-type" formats (e.g. jpeg, mpeg, avi, etc.). Furthermore, one can link to out-of-band data using a URI to a file on the web, a web service request, or an attachment within a SOAP message.

In addition defining a model for observations, the O&M specification also provides a schema for defining phenomena. The O&M *Phenomenon* object (as well as the derived *ConstrainedPhenomenon* and *CompoundPhenomenon*) enables the creation of a dictionary of phenomena. These phenomenon dictionary entrees are utilized as values for the *observable* property in both O&M *Observation* and SensorML *ProcessModel*.

As of February 2006, the O&M specification [OGC 05-087r2] has been submitted for approval as a Best Practices Paper with the intent to immediately begin the RFC process toward final approval as an OGC Technical Specification.

6.1.2 Sensor Model Language (SensorML) [OGC 05-086r2]

The measurement of phenomena that results in an observation consists of a series of processes, beginning with the processes of sampling and detecting and followed perhaps by processes of data manipulation. The division between measurement and "post-processing" has become blurred with the introduction of more complex and intelligent sensors, as well as the application of more on-board processing of observations. The typical Global Positioning System (GPS) sensor is a prime example of a device that consists of basic detectors complemented by a series of complex processes that result in the observations of position, heading, and velocity.

SensorML defines models and XML Schema for describing any process, including measurement by a sensor system, as well as post-measurement processing. SensorML supports a variety of needs within the sensor community, including:

Discovery of sensor, sensor systems, and processes - SensorML is a means by which sensor systems or processes can be functionally described. SensorML provides a rich collection of metadata that can be mined and used for discovery of sensor systems and observation processes.

On-demand processing of Observations - Process chains for geolocation or higher-level processing of observations can be described in SensorML, discovered and distributed over the web, and executed on-demand without a priori knowledge of the sensor or processor characteristics.

Lineage of Observations - SensorML can provide a complete and unambiguous description of the lineage of an observation. In other words, it can describe in detail the process by which an observation came to be, from acquisition by one or more detectors to processing and perhaps even interpretation by an analyst. Not only can this provide a confidence level with regard to an observation, in most cases, part or all of the process could be repeated, perhaps with some modifications to the process or by simulating the observation with a known signature source.

Support for tasking, observation, and alert services - SensorML descriptions of sensor systems or simulations can be mined in support of establishing OGC Sensor Observation Services (SOS), Sensor Planning Services (SPS), and Sensor Alert Services (SAS). SensorML defines and builds on common data definitions that are used throughout the OGC Sensor Web Enablement (SWE) framework.

Plug-N-Play, auto-configuring, and autonomous sensor networks - SensorML enables the development of plug-n-play sensors, simulations, and processes, which seamlessly be added to Decision Support systems. The self-describing characteristic of SensorML-enabled sensors and processes also supports the development of auto-configuring sensor networks, as well as the development of autonomous sensor networks in which sensors can publish alerts and tasks to which other sensors can subscribe and react.

Archiving of Sensor Parameters - Finally, SensorML provides a mechanism for archiving fundamental parameters and assumptions regarding sensors and processes, so that observations from these systems can still be reprocessed and improved long after the origin mission has ended. This is proving to be critical for long-range applications such as global change monitoring and modeling.

Within SensorML, everything including detectors, actuators, filters, and operators are defined as process models. A *ProcessModel* defines the *inputs*, *outputs*, *parameters*, and *method* for that process, as well as a collection of metadata useful for discovery and human assistance. The inputs, outputs, and parameters are all defined using SWE Common data types. Process metadata includes identifiers, classifiers, constraints (time, legal, and security), capabilities, characteristics, contacts, and references, in addition to inputs, outputs, parameters, and system location.

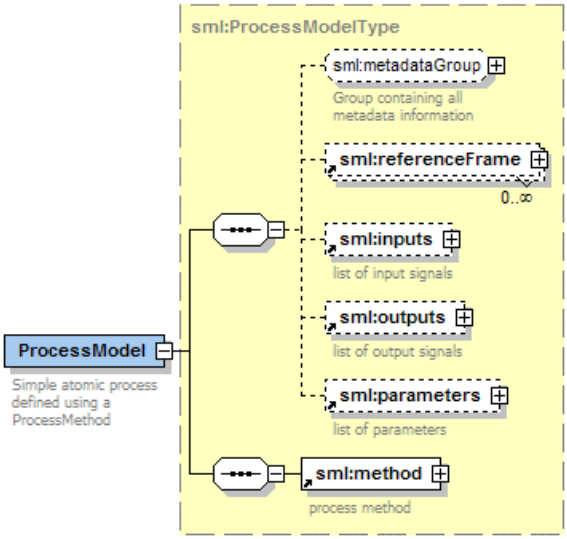


Figure 2. Base properties of the SensorML *ProcessModel* schema, which provides the foundation for atomic process models that can participate in a *ProcessChain* or *System*.

These individual processes, as well as data sources, can the linked within a *ProcessChain* such that one can describe either the process by which an observation was derived (i.e. its lineage) or a process by which additional information can be derived from an existing observation. The SensorML *System* allows one to relate one or more processes to the "real world" by allowing one to specify relative locations and data interfaces.

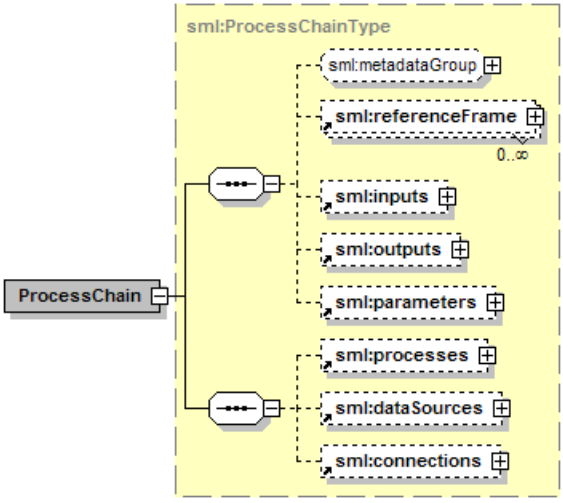


Figure 3. Base properties of the SensorML *ProcessChain* which has metadata, inputs, outputs, and parameters, like a ProcessModel, but can also define sub-processes, data sources, and the connections between these components.

In November 2005, the SensorML specification document OGC 05-086 was approved as a Best Practices Paper. The Request-For-Comment (RFC) process required to approve SensorML as a Technical Specification is currently underway (February 2006). A White Paper document describing the steps for defining and submitting a *ProcessModel* is available and has been submitted to OGC for approval as a Public Discussion Paper. A SourceForge project has been established for submission of *ProcessModel* definitions and supporting software. A public forum for SensorML is actively available at http://mail.opengeospatial.org/mailman/listinfo/sensorml while additional information can be found at http://vast.uah.edu/SensorML.

6.1.3 Transducer Model Language (TML) [OGC 06-010]

Transducer Markup Language (TML) is a method and message format for describing information about transducers and transducer systems and capturing, exchanging, and archiving live, historical and future data received and produced by them. A transducer is a superset of sensors and actuators. TML provides a mechanism to efficiently and effectively capture, transport and archive transducer data, in a common form, regardless of the original source. Having a common data language for transducers enables a TML process and control system to exchange command (control data) and status (sensor data) information with a transducer system incorporating TML technology. TML utilizes XML for the capture and exchange of data.

TML was designed with the express goal of facilitating the development of a "Common" Transducer Processing/Control machine while also facilitating interoperable machine-to-machine communications. For the purposes of data fusion and post analysis, it is paramount to preserve raw transducer data in as close a manner to the original form as possible. Although data would be ideally preserved in its raw format, it is impossible in some cases to do so. TML provides facilities to capture data at any stage, from raw production, to partially processed, to final data forms. Greater benefits of TML are realized the closer to the source raw data one gets.

Transducer Markup Language (TML) defines:

- A set of models describing the hardware response characteristics of a transducer.
- An efficient method for transporting sensor data and preparing it for fusion through spatial and temporal associations

Sensor data is often an artifact of the sensor's internal processing rather than a true record of phenomena state. The effects of this processing on sensed phenomena are hardware-based and can be characterized as functions.

TML response models are formalized XML descriptions of these known hardware behaviors. The models can be used to reverse distorting effects and return artifact values to the phenomena realm. TML provides models for a transducer's latency and integration times, noise figure, spatial and temporal geometries, frequency response, steady-state response and impulse response.

Traditional XML wraps each data element in a semantically meaningful tag. The rich semantic capability of XML is in general better suited to data exchange rather than live delivery where variable bandwidth is a factor. TML addresses the live scenario. The TML cluster is a terse XML envelope designed for efficient transport of live multiplex sensor data. It also provides a mechanism for temporal correlation to other transducer data.

In November 2005, the TML specification document OGC 05-085 was approved as a Public Discussion Paper. In February 2006, TML document OGC 06-010 was submitted as a Pending Document to the Huntsville Technical Committee (March 2006) to begin the RFC process as an OGC Technical Specification. Further information on TML can be found at http://www.transducerml.org .

6.1.4 SWE Common [OGC 05-086r2; OGC 05-087r2]

SWE Common is used throughout the SWE framework and provides a "common" means to specify expected or observed data components[2]. SWE Common elements are used to construct machine-readable descriptions of data, encodings and values.

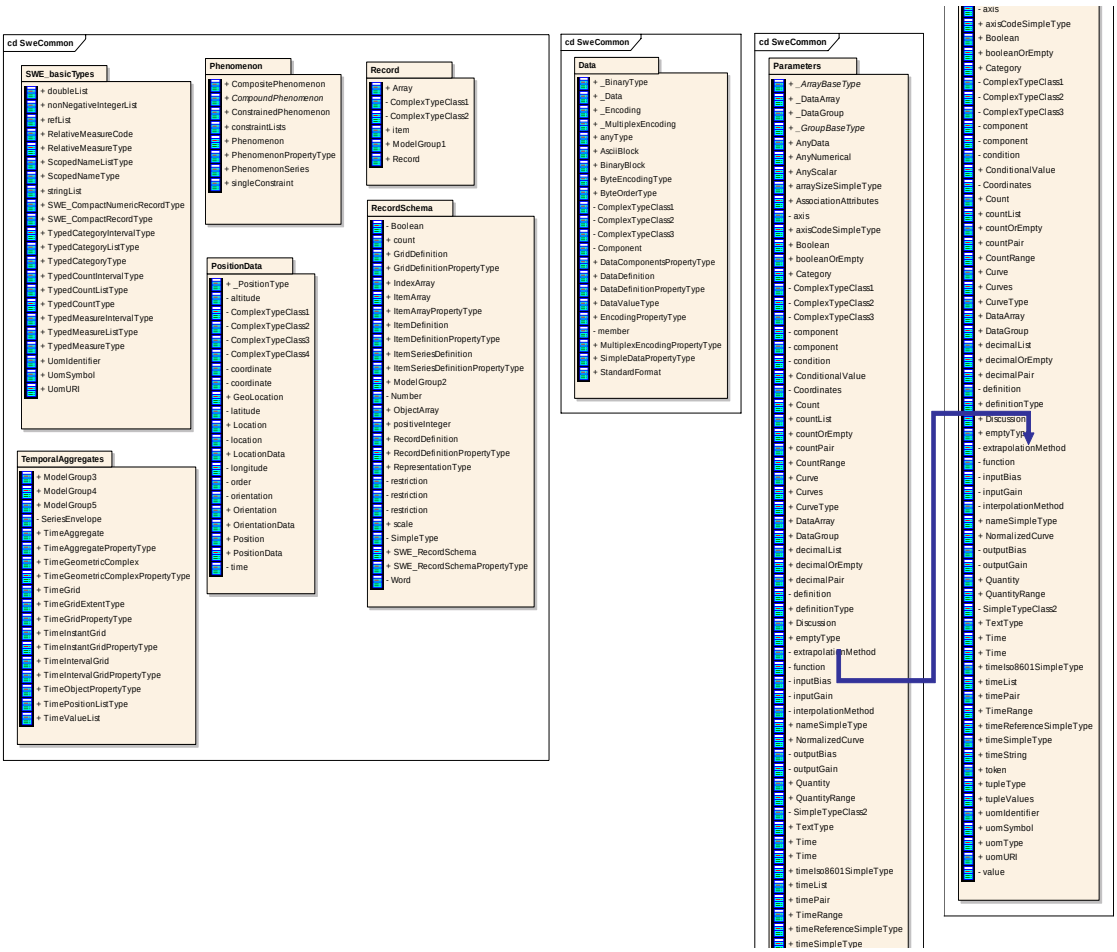


Figure 4. SWE Common Components

SWE Common is comprised of the schema components shown in Figure 4 and described here:

- **BasicTypes**: building block type definitions for lists, records, categories, counts, measures and units of measure.

- **TemporalAggregates**: basic types for describing temporal positions, ranges, extents, values, aggregations and their properties.

[2] SWE Common elements are currently not specified within a single document but rather within the SensorML and O&M specification documents where they are used. Future versions may separate SWE Common definitions into their own specification document.

- **Phenomenon**: elements for identifying and describing phenomena used for soft-typing of property elements.

- **Position**: elements for characterizing location and orientation in spatio-temporal reference frames (e.g., geodetic and local "engineering" coordinate systems) as well as state properties such as velocity and acceleration.

- **Record** and **RecordSchema**: building blocks for describing and encoding axis, record and grid data structure components (e.g., item, length, dimension, series, tuple, array, map, property, and property-set) used for soft-typing of property elements.

- **Data**: elements used to explicitly describe data components and their values, allowing flexibility for encoding large amounts of data in compact forms (e.g., supporting efficient ASCII and Binary data blocks as well as MIME types).

- **Parameters**: building blocks for describing and encoding parameter constructs such as scalar (e.g., quantity, count, boolean, category and time), curve, tuple, unit of measure and temporal values.

These fundamental data types are used and extended by SWE specifications and applications. For example, the SensorML specification uses the SWE Common elements as the fundamental data types for all *input*, *output*, and *parameter* definitions, as well as property values within *characteristics* and *capabilities*.

Figure 5 depicts the general form of SWE Common data definition, encoding and value elements for constructing SWE system parameters, inputs and outputs. Note that data components (e.g., quantity, count, boolean category and time) definitions can be optionally grouped to define higher order data constructs (e.g., weather measurements comprised of temperature, precipitation, and wind speed). Note also that a Data Definition may occur without a data value element, the definition by itself having semantic utility in its own right.

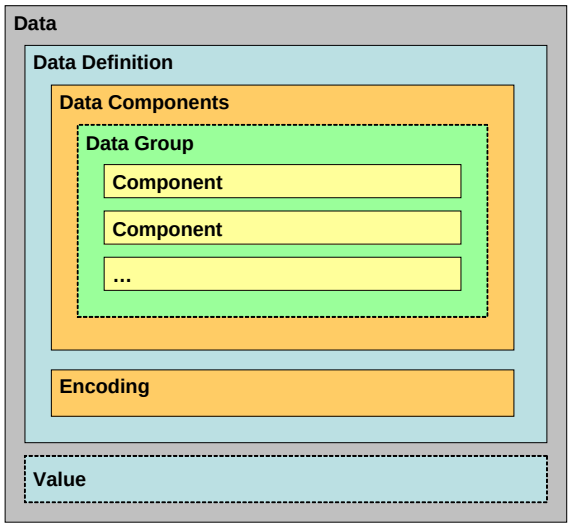


Figure 5. SWE Common Data Model

We present below several examples of SWE Common data definitions as XML instances. In the first listing, SWE Common elements are used within a SensorML document to describe *outputs*. For example, the output of a GPS sensor might include a GPS status data cluster described as the following *DataGroup*:

```
<output name="status">
    <swe:DataGroup>
        <swe:component name="hFOM">
            <swe:Time definition="urn:ogc:data:ogc1.0.3:gps:HFOM" uom="urn:ogc:unit:meter"/>
        </swe:component>
        <swe:component name="vVFOM">
            <swe:Quantity definition="urn:ogc:data: ogc:.0.3:gps:VFOM" uom="urn:ogc:unit:meter"/>
        </swe:component>
        <swe:component name="hDOP">
            <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:gps:HDOP" uom="urn:ogc:unit:meter"/>
        </swe:component>
        <swe:component name="vDOP">
            <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:gps:VDOP" uom="urn:ogc:unit:meter"/>
        </swe:component>
        <swe:component name="numSys">
            <swe:Count definition="urn:ogc:data:ogc:1.0.3:gps:NUMSYS"/>
        </swe:component>
        <swe:component name="navMode">
            <swe:Count definition="urn:ogc:data:ogc:1.0.3:gps:NAVMODE"/>
        </swe:component>
    </swe:DataGroup>
</output>
```

Similarly, the parameters of a thermometer described in SensorML might include a calibration curve defined using SWE Common data components:

```
<parameters>
    <ParameterList>
        <steadyStateResponse>
            <NormalizedCurve fixed="true">
                <function>
                    <swe:Curve arraySize="21">
```

```
                        <swe:definition>
                            <swe:Coordinates>
                                <swe:axis name="temperature">
                                    <swe:Quantity id="TEMP1"
                                        definition="urn:ogc:def:phenomenon:ogc:1.0.3:temperature"
                                        uom="urn:ogc:unit:celsius"/>
                                </swe:axis>
                                <swe:axis name="resistance">
                                    <swe:Quantity id="RES"
                                        definition="urn:ogc:def:phenomenon:ogc:1.0.3:resistance"
                                        uom="urn:ogc:unit:ohm" scale="1e3"/>
                                </swe:axis>
                            </swe:Coordinates>
                        </swe:definition>
                        <swe:tupleValues>-40,328.4 -35,237.7 -30,173.9 -25,128.5 -20,95.89 -15,72.23
                            -10,54.89 -5,42.07 0,32.51 5,25.31 10,19.86 15,15.69 20,12.49 25,10 30,8.06
                            35,6.536 40,5.331 45,4.373 50,3.606 55,2.989 60,2.49
                        </swe:tupleValues>
                    </swe:Curve>
                </function>
            </NormalizedCurve>
        </steadyStateResponse>
    </ParameterList>
</parameters>
```

Likewise, *characteristics* and *capabilities* within SensorML utilize SWE Common data components as in:

```
<capabilities>
    <PropertyList>
        <property name="measurementProperties">
            <MeasurementCapabilities>
                <measureResolution>
                    <swe:Quantity definition="urn:ogc:def:phenomenonogc:1.0.3:temperature"
                        uom="urn:ogc:def:unit:ogc:1.0:degreeCelsius"> 0.1 </swe:Quantity>
                </measureResolution>
                <dynamicRange>
                    <swe:QuantityRange definition="urn:ogc:def:phenomenonogc:1.0.3:temperature "
                        uom=" urn:ogc:def:unit:ogc:1.0:degreeCelsius "> -45 60 </swe:QuantityRange>
                </dynamicRange>
                <accuracy>
                    <swe:QuantityRange definition="urn:ogc:def:data:ogc:1.0.3:accuracy"
                        uom="urn:ogc:def:unit:ogc:1.0:percent"> -0.5 0.5 </swe:QuantityRange>
                </accuracy>
            </MeasurementCapabilities>
        </property>
        <property name="survivableRange">
            <Limits definition="urn:ogc:def:property:ogc:1.0.3:survivableLimits">
                <limit name="windSpeedLimits">
                    <swe:QuantityRange definition="urn:ogc:def:phenomenon:ogc:1.0.2:windSpeed"
                        uom="urn:ogc:unit:metersPerSecond">0 175</swe:QuantityRange>
                </limit>
            </Limits>
        </property>
    </PropertyList>
</capabilities>
```

Within the O&M CommonObservation, the SWE Common data definitions provide a complete description of the resulting *Data*, including the *swe:dataComponents* and the *swe:encoding*:

```
<swe:Data>
    <swe:definition>
```

```
            <swe:DataDefinition>
                <swe:dataComponents>
                    <swe:DataGroup>
                        <swe:component name="time">
                            <swe:Time definition="urn:ogc:def:phenomenon:ogc:1.0.3:time"
                                referenceTime="1970-01-01T00:00:00Z" uom="urn:ogc:unit:second"/>
                        </swe:component>
                        <swe:component name="latitude">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                                uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                        </swe:component>
                        <swe:component name="longitude">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                                uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                        </swe:component>
                        <swe:component name="altitude">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:distance"
                                 uom="urn:ogc:def:unit:ogc:1.0:foot"/>
                        </swe:component>
                        <swe:component name="trueHeading">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                                uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                        </swe:component>
                        <swe:component name="pitch">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                                uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                        </swe:component>
                        <swe:component name="roll">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                                uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                        </swe:component>
                    </swe:DataGroup>
                </swe:dataComponents>
                <swe:encoding>
                    <swe:AsciiBlock decimalSeparator="." tokenSeparator="," tupleSeparator=" "/>
                </swe:encoding>
            </swe:DataDefinition>
        </swe:definition>
        <swe:value>
            1.068088738124E9,32.6018,-116.6153,13790,0.0,7.218017578125,-0.999755859375
            1.068088738124E9,32.6018,-116.6153,13790,0.0,7.174072265625,-1.0986328125
            1.068088738124E9,32.6018,-116.6153,13790,0.0,7.1685791015625,-1.16455078125
            1.068088738124E9,32.6018,-116.6153,13790,0.0,7.152099609375,-1.20849609375
            1.068088738124E9,32.6018,-116.6153,13790,0.0,7.14111328125,-1.219482421875
            1.068088738124E9,32.6018,-116.6153,13790,0.0,7.174072265625,-1.241455078125
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.174072265625,-1.2469482421875
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.2235107421875,-1.2469482421875
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.2235107421875,-1.23046875
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.2454833984375,-1.1920166015625
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.283935546875,-1.131591796875
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.305908203125,-1.087646484375
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.327880859375,-1.0382080078125
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.327880859375,-0.955810546875
            1.068088739124E9,32.6017,-116.6161,13790,0.0,7.3333740234375,-0.90087890625
        <swe:value>
    </swe:Data>
```

Similarly, SWE Common data definitions are used to describe sensor tasking input within the Sensor Planning Service:

```
<DescribeTaskingRequestResponse>
    <taskingDescriptor>
        <sensorID>ifgicam</sensorID>
        <InputDescriptor parameterID="zoom" updateable="true" use="optional"
                xmlns:sps1="http://www.opengis.net/sps">
```

```
            <gml:description xmlns:gml="http://www.opengis.net/gml">Zoom the device n steps.</gml:description>
            <sps1:definition>
                <swe:DataDefinition xmlns:swe="http://www.opengis.net/swe">
                    <swe:dataComponents>
                        <swe:Count max="9999" min="1"/>
                    </swe:dataComponents>
                    <swe:encoding>
                        <swe:StandardFormat mimeType="text"/>
                    </swe:encoding>
                </swe:DataDefinition>
            </sps1:definition>
        </InputDescriptor>
        <InputDescriptor parameterID="pan" updateable="true" use="optional"
            xmlns:sps1="http://www.opengis.net/sps">
            <gml:description xmlns:gml="http://www.opengis.net/gml">Pan the device relative to the (0,0)
                position.</gml:description>
            <sps1:definition>
                <swe:DataDefinition xmlns:swe="http://www.opengis.net/swe">
                    <swe:dataComponents>
                        <swe:Quantity max="180.0" min="-180.0"/>
                    </swe:dataComponents>
                    <swe:encoding>
                        <swe:StandardFormat mimeType="text"/>
                    </swe:encoding>
                </swe:DataDefinition>
            </sps1:definition>
        </InputDescriptor>
        <InputDescriptor parameterID="task-start-time" updateable="false" use="required"
            xmlns:sps1="http://www.opengis.net/sps">
            <gml:description xmlns:gml="http://www.opengis.net/gml">Give the start-time of your task as one
                dateTime-instance encoded as ISO8601. </gml:description>
                <sps1:definition>
                    <swe:DataDefinition xmlns:swe="http://www.opengis.net/swe">
                        <swe:dataComponents>
                            <swe:IsoDateTime/>
                        </swe:dataComponents>
                        <swe:encoding>
                            <swe:StandardFormat mimeType="text"/>
                        </swe:encoding>
                    </swe:DataDefinition>
                </sps1:definition>
            </InputDescriptor>
        </taskingDescriptor>
    </DescribeTaskingRequestResponse>
```

Also within the SWE Common namespace (http://www.opengis.net/swe) are definitions for describing a *Phenomenon*, including those for simple, compound, and constrained phenomena. The common definitions for data components, encoding, and phenomenon are currently used throughout every SWE component with the exception of TML and SAS. Efforts during OWS3 have illustrated that both of these specifications can be supported using the SWE Common definitions. It is anticipated that future efforts will better support the use of SWE Common schema components throughout the entire SWE framework, including TML and SAS.

6.2 Service Model

6.2.1 Sensor Observation Service (SOS) [OGC 06-009]

The goal of SOS is to provide access to observations from sensors and sensor systems in a standard way that is consistent for all sensor systems including remote, in-situ, fixed

and mobile sensors. This is a challenging task because the users of sensor data have historically been divided into those who primarily deal with in-situ sensors and those who primarily deal with remote sensors. The terminology, perspective, and expectations of these two broad groups are different. SOS leverages the Observation and Measurements (O&M) specification for modeling sensor observations and the SensorML specification for modeling sensors and sensor systems.

An SOS organizes collections of related sensor system observations into Observation Offerings. The concept of an Observation Offering is equivalent to that of a sensor constellation discussed earlier in this document. An Observation Offering is also analogous to a "layer" in Web Map Service because each offering is intended to be a spatially and/or thematically non-overlapping group of related observations. Each Observation Offering is constrained by a number of parameters including the following:

- Specific sensor systems that report the observations,
- Time period(s) for which observations may be requested (supports historical data),
- Phenomena that are being sensed,
- Geographical region that contains the sensors, and
- Geographical region that is the subject of the sensor observations (may differ from the sensor region for remote sensors)

The following table lists all operations defined for the SOS interface:

Operation Name	Arguments	Description
GetCapabilities GET (POST)	service (='SOS'), version, sections	Used to retrieve service metadata.
GetObservation POST (GET)	service, version, offering, procedure, observedProperty, eventTime, result featureOfInterest, resultFormat, resultModel, responseMode	Allows client to request archived or real time observation data [om:Observation]
DescribeSensor POST (GET)	service, version, sensorID, outputFormat	Allows client to retrieve the sensor description [SensorML:System] associated with a given offering
RegisterSensor (POST)	service, version, SensorML system, O&M observation template	Allows the client to register a new sensor system with the SOS
InsertObservation (POST)	service, version, insertID, O&M observation	Allows the client to insert new observations for a sensor system
GetResult (POST) (GET)	service, version, observationID, timeInstant, geometricPrimitive	Allows a client to repeatedly obtain sensor data from the same set of sensors
GetFeatureOfInterest (POST) (GET)	service, version, featureOfInterestID, eventTime, featureOfInterestLocation	Allows a client to request the complete description of the feature of interest associated with

		an observation offering
GetFeatureOfInterestTime (POST) (GET)	service, version, objectID, spatialOps	Returns time periods for which the SOS will return data for an advertised feature of interest
DescribeFeatureOfInterest (POST) (GET)	service, version, featureOfInterestID	Returns the XML schema for the specified GML feature of interest advertised in the capabilities
DescribeObservationType (POST) (GET)	service, version, phenomenon	Returns the XML schema that describes the Observation type returned by GetObservation
DescribeResultModel (POST)(GET)	service, version, resultName	Returns the XML schema for the result element returned by GetObservation

The approach that has been taken in the development of SOS, and the SWE specifications on which it depends, is to carefully model sensors, sensor systems, and observations in such a way that the model covers all varieties of sensors and supports the requirements of all users of sensor data. This is in contrast to the approach that was taken with the Web Feature Service (WFS). WFS provides an interface for requesting feature instances or collections where the feature-type is not constrained by the WFS specification but is described at the service instance level where the formalization of the feature definition is as a GML application schema. With this approach, interoperability requires organizations to agree on domain-specific GML application schemas and to formulate WFS requests using the generic Filter syntax. Clients that access WFS instances in a particular domain must have a-priori knowledge of the application schemas used in that domain. The SOS approach defines a common model for all sensors, sensor systems and their observations. This model is not domain-specific and can be used without a-priori knowledge of domain-specific application schemas.

Note that, since Observation can be implemented as a GML feature-type, it would be possible to deliver Observations through a WFS. However, it is convenient to define SOS as a specialized interface for this restricted feature type (and specializations thereof) since this allows some of the properties of the observation feature type to be promoted and to have specific slots in the request syntax that would be more laborious to express using the generic Filter syntax of the generic WFS.

In November 2005, the SOS specification document OGC 05-088r1 was approved as a Public Discussion Paper. In February 2006, SOS document OGC 06-009 was submitted as a Pending Document to the Huntsville Technical Committee (March 2006) to begin the RFC process toward approval as an OGC Technical Specification.

6.2.2 Sensor Alert Service (SAS) [OGC 05-098]

The Sensor Alert Service (SAS) can be compared with an event notification system. An SAS might therefore provide a wide variety of alerts related to sensors and sensor observations including, as examples, measured values above a threshold, detected motion

or the presence of a recognizable feature, or perhaps sensor status (e.g. low battery, shutdown or startup).

An SAS can *advertise* what alerts it can provide. A consumer (interested party) may *subscribe* to alerts disseminated by the SAS. If an event occurs the SAS will *publish* an alert and *notify* all clients subscribed to this event type through a messaging service, such as WNS.

The specified SAS specifies an interface that allows measurement or processing systems to advertise and publish observational data or its describing metadata respectively. It is important to emphasize that the SAS itself acts rather like a registry than an event notification system! A messaging server (which can be on the same server as the SAS or separate) is indeed responsible for issuing alerts. The table below shows all operations defined by the SAS interface:

Operation Name	Arguments	Description
GetCapabilities GET (POST)	service (=‘SAS’), version	Used to retrieve service metadata.
Publish POST	service, version, alert content	Allows a producer to publish its alerts
Advertise POST	service, version, systemDescriptionEndpoint, result alertFormat, advertisementID , alertTime, observedProperty, featureOfInterest, desiredPublicationExpiration transportBinding, subscribeEndpoint	Allows a producer to advertise its alerts
CancelAdvertisement POST (GET)	service, version, publicationID	Allows producers to cancel an advertisement
RenewAdvertisement POST (GET)	service, version, publicationID, desiredPublicationExpiration	Allows producers to renew an advertisement
Subscribe POST (GET)	service, version, subscriptionOfferingID resultFormat, transportbinding alertTime, procedure, observedProperty, featureOfInterest, resultDefinition, desiredSubscriptionExpiration	Allows consumers to subscribe to alerts and define custom alerts
CancelSubscription POST (GET)	service, version, subscriptionID	Allows consumers to cancel a subscription
RenewSubscription POST (GET)	service, version, subscriptionID, desiredSubscriptionExpiration	Allows consumers to renew a subscription

Traditional OGC web services are not suitable for implementing this alert service. Instead of regular request/response protocols such as HTTP, the XMPP protocol was widely used

during our initial tests, but it is important to emphasize that the SAS specification is agnostic regarding the alert protocol used.

The SAS specification has been developed under an Interoperability Experiment. While it was not a part of the OWS3 efforts, although it shared many common participants. The current SAS specification document [OGC 05-098] is published as a Draft Interoperability Program Report.

6.2.3 Sensor Planning Service (SPS) [OGC 05-089r2]

The Sensor Planning Service (SPS) is intended to provide a standard interface to collection assets (i.e., sensors, and other information gathering assets) and to the support systems that surround them. Not only must different kinds of assets with differing capabilities be supported, but also different kinds of request processing systems, which may or may not provide access to the different stages of planning, scheduling, tasking, collection, processing, archiving, and distribution of requests and the resulting observation data and information that is the result of the requests. The SPS is designed to be flexible enough to handle such a wide variety of configurations. The following table lists all operations defined on the SPS interface:

Operation Name	Arguments	Description
GetCapabilities GET (POST)	service (=‘SPS’), version, sections	Used to retrieve service metadata.
DescribeTasking POST (GET)	service, version, sensorID	Allows a client to request all parameters that have to be set to perform a Submit operation
GetFeasibility (POST)	service, version, notificationTarget, sensorID, parameters, timeFrame	Allows to request feedback about the feasibility of a tasking request
Submit POST	service, version, notificationTarget, sensorParam, feasibilityID, timeFrame	Allows client to submit the assignment request
GetStatus (POST) (GET)	service, version, notificationTarget, taskID	Allows producers to renew an advertisement
Update (POST)	service, version, notificationTarget, taskID, parameters	Allows a client to update a previously submitted task
Cancel (POST) (GET)	service, version, taskID	Allows a client to cancel a previously submitted task
DescribeResultAccess POST (GET)	service, version, taskID, sensorID	Allows consumers to renew a subscription

The SPS specification document [OGC 05-089r2] was approved as an OGC Public Discussion Paper in Bonn in November 2005.

6.2.4 Web Notification Service (WNS) [OGC 05-114]

Web Service environments provide a suitable method to gather requested information in an appropriate way. Synchronous transport protocols such as HTTP provide the necessary functionalities to post requests and to receive the respective responses. HTTP is a reliable protocol in the way it ensures the packet delivery, in order, and with a definitive acknowledgement for each delivery or failure. In case of a simple Web Map Service, a user will receive visualized geographic information after a negligible amount of time, or the user will receive an exception message. HTTP satisfies the needs for this kind of processing, without the need for further functionality.

As services become more complex, basic request-response mechanisms need to contend with delays/failures. For example, mid-term or long-term (trans-) actions demand functions to support asynchronous communications between a user and the corresponding service, or between two services, respectively. A Web Notification Service (WNS) is required to fulfill these needs within the SWE framework.

The Web Notification Service Model includes two different kinds of notifications. First, the “one-way-communication” provides the user with information without expecting a response. Second, the “two-way-communication” provides the user with information and expects some kind of asynchronous response. This differentiation implies the differences between simple and sophisticated WNS. A simple WNS provides the capability to notify a user and/or service that a specific event occurred. In addition, the latter is able to receive a response from the user.

The basis on which notifications will be sent is free to the service and will be described in its capabilities. The “way-of-notification” palette may include:

- e-mail
- http-call (as HTTP POST: in case of sophisticated clients that act as web services themselves)
- SMS
- Instant Message
- phone call
- letter
- fax

Once a client registers a user along with the method of notification desired, the client receives a unique RegistrationID that can then be provided as input to other services (e.g. SPS or SAS).

Operation Name	Arguments	Description
GetCapabilities GET (POST)	service (='WNS'), version, sections	Used to retrieve service metadata.
registerUser POST	Name of user, communication protocol desired	Allows a client to register a user for future notification; returns a unique user RegistrationID
doNotification POST	User ID, process ID, message containing status of operation requested (e.g. operation completed, operation failed, etc)	provides status of operation requested; returns notification status indicating the success of the notification sending procedure
doCommunication POST	User ID, processID, callback interface, and message describing additional paramater values required to complete requested operation	Requests additional information; returns notification status indicating the success of the notification sending procedure
doReply POST	User ID, process ID, message containing values for the required parameters	Allows user to response to a request for more information; returns status indicating the success of forwarding this reply to the calling service

The WNS specification document [OGC 05-114] was approved as a OGC Public Discussion Paper in early 2005.

6.2.5 Sensor Web Registry

The Sensor Web registry was implemented successfully using an OGC Catalog Service backed up by an ebRIM/ebXML engine. This service provides discovery capability throughout the whole sensor web infrastructure. Typical requests to this service are 'GetRecords' operations containing filtering parameters used to search a database for one or more matching objects of interest. These objects include SWE services (as well as other OGC services), sensor descriptions, process chains and dictionary entries such as phenomena or units, etc. The following table summarizes available CSW/HTTP operations (optional bindings are shown in parenthesis):

Operation Name	Arguments	Description
GetCapabilities GET (POST)	service (='CSW'), version (='2.0.0')	Used to retrieve service metadata.
DescribeRecord POST (GET)	service, version, namespace, typeName, outputFormat, schemaLanguage	Used to discover elements of the information model supported by the target catalogue service
GetDomain POST (GET)	service, version, parameterName, propertyName	Used to obtain runtime information about the range of values of a metadata record element or request parameter
GetRecords POST (GET)	service, version, namespace, resultType, outputFormat, outputSchema, startPosition, maxRecords, typeNames,	Used for resource discovery

	elementName, constraint, sortBy, distributedSearch, hopCount	
GetRecordById GET (POST)	service, version ,elementSetName, id	Used to retrieve the default representation of catalogue records using their identifier
Transaction POST	handle, action (insert, delete, update), requestId, verboseResponse	Used for creating, modifying and deleting catalogue records (push)
Harvest POST	service, version, namespace, source, resourceType, resourceFormat, responseHandler, harvestInternal	Used to insert objects into the catalogue by providing a link to the data to be added (pull mechanism)

In order to be able to insert objects to a catalogue, each object type must be defined by a schema and a CSW harvest profile. This profile shall define what information needs to be parsed out of the object XML and advertised as searchable content.

The following table shows what pieces can be mined from different XML documents used through the SWE framework:

Document Name	**Searchable Sections/Tags**
SOS Capabilities	OWS common section (like any other service) For each observation in the offering list: - observation id, name and description - observed property (association with O&M phenomenon object) - procedure id (association with SensorML sensor object) - feature of interest (association with GML feature) - time range - location (if fixed) - format
SPS Capabilities	OWS common section (like any other service) For each sensor system in the offering list: - phenomenon urn (association with O&M phenomenon object) - sensor id (association with SensorML sensor object) - area of service
SAS Capabilities	OWS common section (like any other service) For each subscription in the offering list: - alert id, name and description - observed property (association with O&M phenomenon object) - procedure id (association with SensorML sensor object) - feature of interest (association with GML feature) - time range - location (if fixed) - format
SensorML Sensor, System and Process	Most information is contained in the metadata group - description - identifiers - classifiers - time, legal and security constraints - characteristics

	- capabilities - contacts - inputs and outputs (association with O&M phenomenon) - taskable parameters (association with O&M phenomenon) ➔ eventually recurse for each sub components
O&M Phenomena	A phenomenon is intended to be a pure dictionary entry, so it should be parsed in its entirety, including: - description - name - base phenomenon (association with other O&M phenomenon) - constraint phenomenon (association with other O&M phenomenon) - constraint value - component if composite (association with other O&M phenomenon)

7 Interrelationships

The SWE architecture is comprised of XML schema defining data and metadata encodings and service interface definitions. The schema packages comprising SWE and their dependencies are shown in Figure 6. The core schema packages comprising SWE are highlighted in yellow.

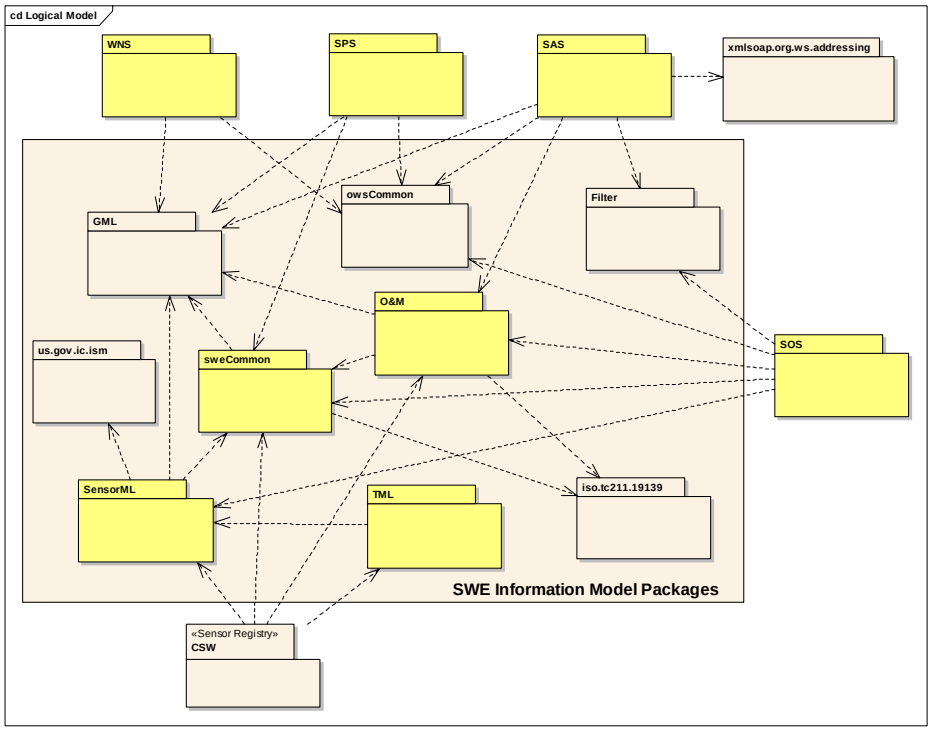


Figure 6. SWE Schema Package Dependencies

7.1 Dependencies on and references to SWE Common

As discussed previously, almost all components within the SWE architecture utilize directly or indirectly schema components defined within the SWE Common namespace (swe). The following maps dependencies on and references to the SWE Common parameter.xsd, data.xsd, and phenomenon.xsd schemas:

The parameter.xsd schema defines basic data types (e.g. *swe:Quantity*, *swe:Count*, *swe:Boolean*, *swe:Category*, and *swe:Time*) as well as data composites (e.g. *swe:DataGroup*, *swe:DataArray*, and *swe:Curve*).

- SensorML: uses data types and aggregates for defining *sml:inputs*, *sml:outputs*, and *sml:parameters*
- SensorML: uses data types and aggregates for defining *sml:characteristics* and *sml:capabilities*

- O&M: uses data types and aggregates for defining *om:dataComponents* within *om:CommonObservation*
- SPS: uses data types and aggregates to define *sps:taskingDescriptor*
- SOS: used in SOS to define *ObservationOffering*
- SAS: used to define Alert structure

- [data.xsd]
  - O&M: used to define Encoding in CommonObservation
  - SensorML: used to define data interface input/output

- [phenomenon.xsd]
  - O&M: used to define "observable" in Observation
  - SensorML: references to Phenomenon entries used in definitions of inputs and outs in sml:Process

7.2 Dependencies on and references to SensorML

- SAS: sensor Alert components reflect the output of sensor or process
- SAS: used in *DescribeSensor*
- SPS: sml:*System* "parameters" provide possible sensor settings that can be tasked in SPS
- SensorRegistry: sml:identifiers", "sml:classifiers", "sml:capabilities", "sml:characteristics" mined for discovery purposes
- SOS: sos:getSensor() returns sml:System
- TML: sml:metadataGroup used in tml:System

7.3 Dependencies on and references to Observations

- SensorML: sml:DataSource can reference om:data in om:CommonObservation
- SOS: is the output of a sos:getObservation() request
- SOS: CommonObservation "dataDefinition" provides result
- ObservationRegistry*: om:Observation* "*time*", "*boundedBy*", "*featureOfInterest*" properties, "*observable*" mined for discovery purposes

7.4 Dependencies on and references to TML

- O&M: *om:CommonObservation* can link to a TML stream
- SOS: optional output of *sos:getObservation* request via *om:CommonObservation*.

7.5 Dependencies on and references to Geography Markup Language (GML)

- SensorML: use *gml:EngineeringCRS* for *sml:referenceFrame* property
- O&M: *om:Observation* is derived from *gml:FeatureType* and utilizes GML property values throughout

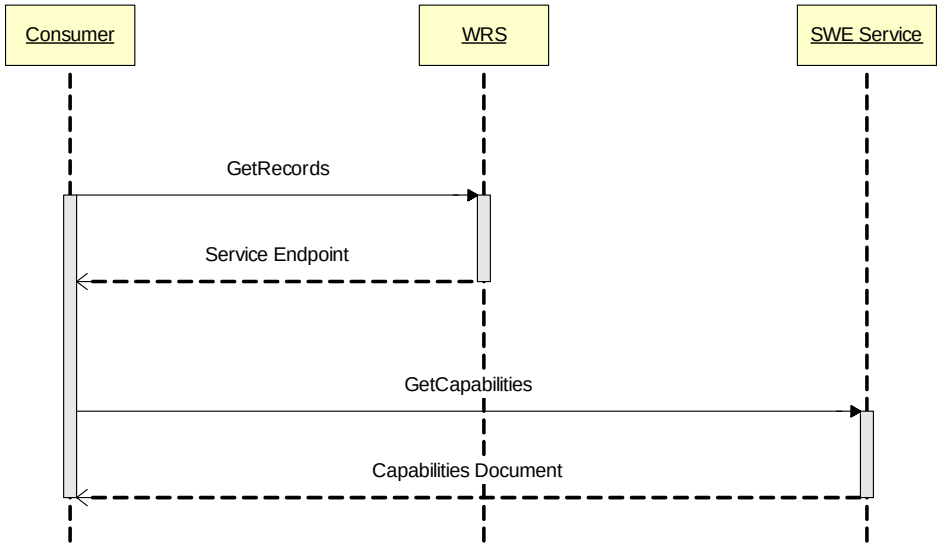
- SOS: can take *gml:Location* as input to *getObservation()* operation
- SPS: can take *gml:Location* as input to *getFeasibility()* operation

8 Typical Work Flows

The following diagrams illustrate simple workflows involving one or several SWE services. This section covers service discovery and registration with a registry, access to discrete and streaming sensor data, sensor tasking, publication of alerts and subscription to alert services. More advanced diagrams show a specific SWE configuration using simple sensor nodes connected via a private network to a data center where data can be archived, processed and made available on the public Internet.

8.1 Discovery of SWE Services using a Registry

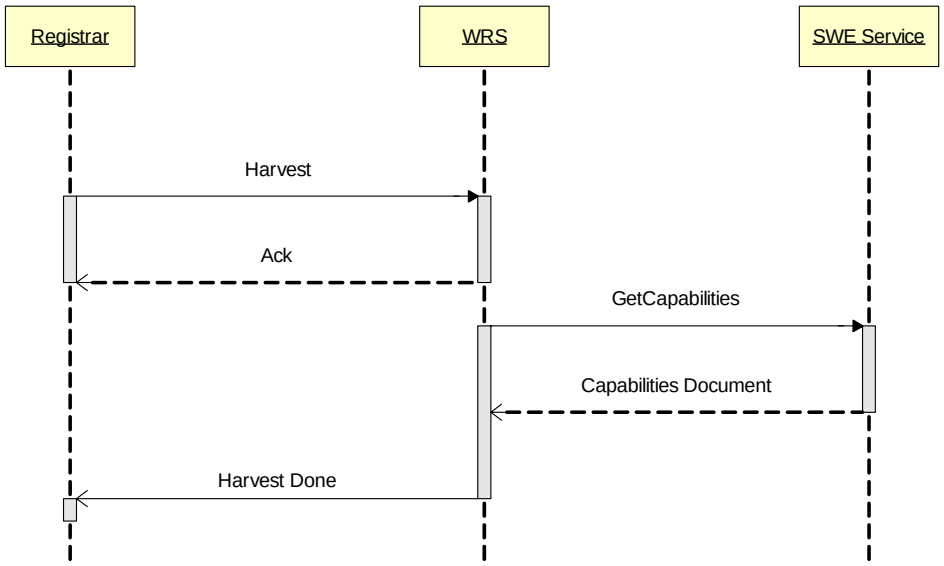
This first scenario shown below demonstrates how SWE services, like any other OGC web service can be discovered using a registry. In this case, the consumer (which can be a client or another service) first connects to a WRS and sends a search query for a specific type of SWE service using a 'GetRecords' message. The WRS looks for a matching record in its database and returns an XML document containing the endpoint URL of all matching services found. The client is then able to connect to the service of interest and request the complete capabilities document.



SWE Service discovery sequence

8.2 Registration of a new SWE Service in a Registry

The following scenario shows how to register a new service to a registry. The registrar, which can be the service itself or a third party, sends a 'Harvest' command to the WRS containing the endpoint URL of the service to be registered. The 'Harvest' operation is then executed asynchronously by the WRS when resources are available. When so, the WRS connects to the specified service, fetches its capabilities document, and starts processing it according to the CSW profile defined for this type of registry object. This profile defines what part of the document should be parsed and used as searchable fields in the registry. When the 'Harvest' operation is complete, the WRS sends a notification message to the registrar. The newly added service is now searchable through the WRS interface.



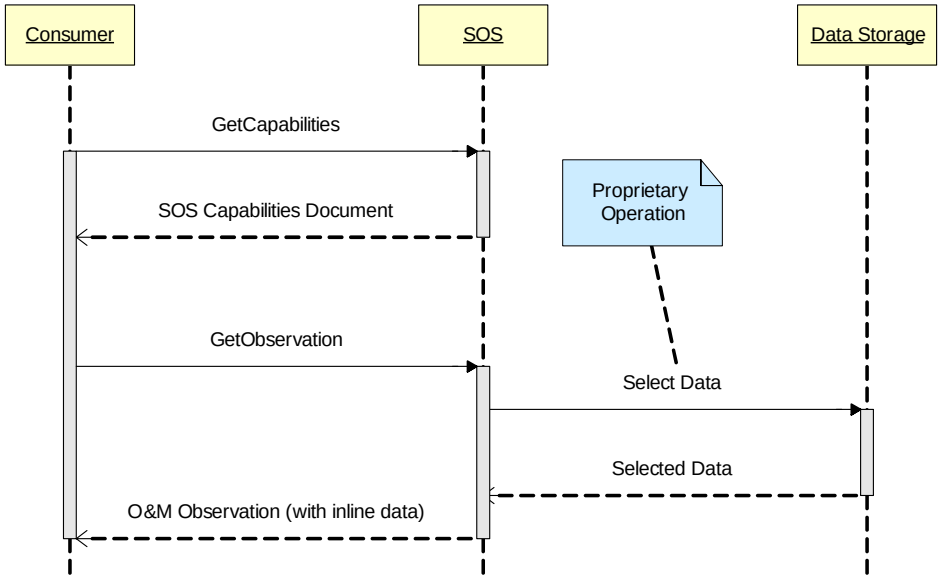
SWE Service registration sequence

(Note that the Registrar can be on the same node as the Service itself)

It is important to realize that each service type should provide a CSW profile in order to be successfully registered by a WRS. A default profile can nevertheless be created using only the common OWS section used by all new OGC services. This will provide a minimum set of searchable information, put doesn't allow to discovery of individual layers or offerings that are service specific.

8.3 Request of discrete observation data

This sequence diagram shows the simple process of requesting observation data from an SOS service. The consumer (perhaps after having discovered the service) issues a 'GetCapabilities' request to obtain a list of all observation offerings available on this server. Using this information, the consumer then selects a specific offering, one or more phenomena, a finite time range and perhaps a region of interest (bbox) and sends a 'GetObservation' command with these parameters to the SOS. The SOS then internally reads the requested data using proprietary technology (perhaps a set of files or a DB) and sends it back to the consumer encapsulated in an O&M observation, which provides metadata as well as temporal and spatial characteristics of the data.

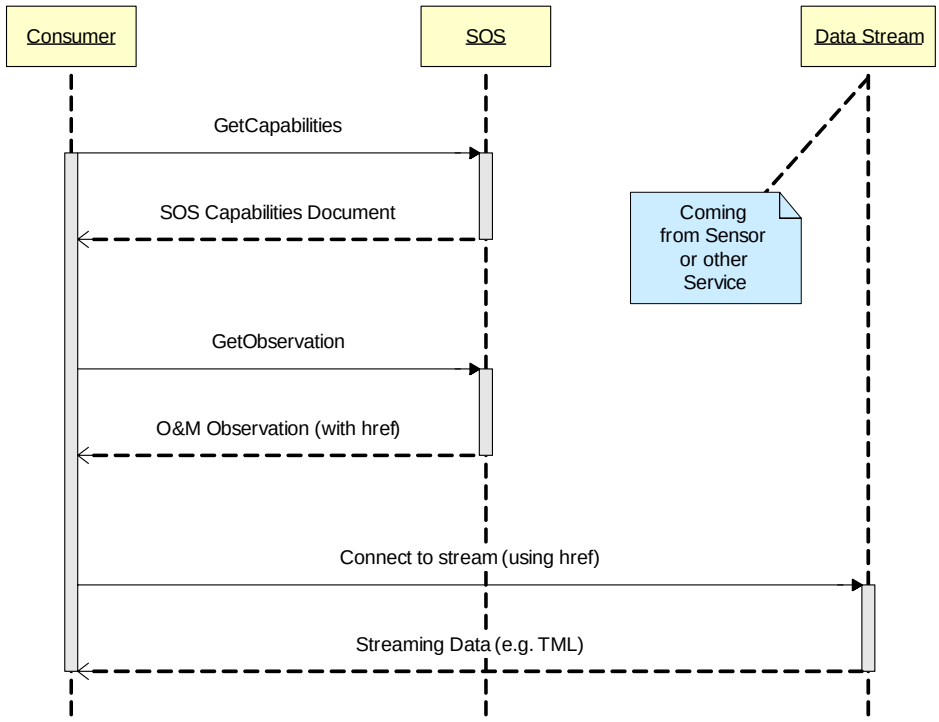


SOS GetObservation for archived data

In this example, the request is based on a finite time range selecting archive data. The size of the block of data is therefore known and it is possible to include it inline within the O&M document. The specially designed SWE 'CommonObservation' (see §6.1) allows the structure and encoding of the data to be fully described, thus allowing base64 encoded data within the XML in order to support heavy datasets such as imagery or radar data (Note that raw binary is also available if providing out of band data).

8.4 Request of streaming (out of band) observation data

This diagram demonstrates how streaming data (eventually real time) can also be accessed through an SOS interface. The consumer first downloads the capabilities document which may contain an offering tagged as real time data. If this is the case, the consumer can issue a 'GetObservation' request for this offering and a time range specifying an end date in the future. The consumer can then start receiving real time data from the SOS or even directly from the source as indicated in the following diagram. In order to achieve this, the data is not provided inline in the O&M observation like in the previous example, but rather through a hyperlink (href) to the data source. This hyperlink can be a call to the SOS 'GetResult' operation which would return only the data, or another (proprietary) static URL or service request.

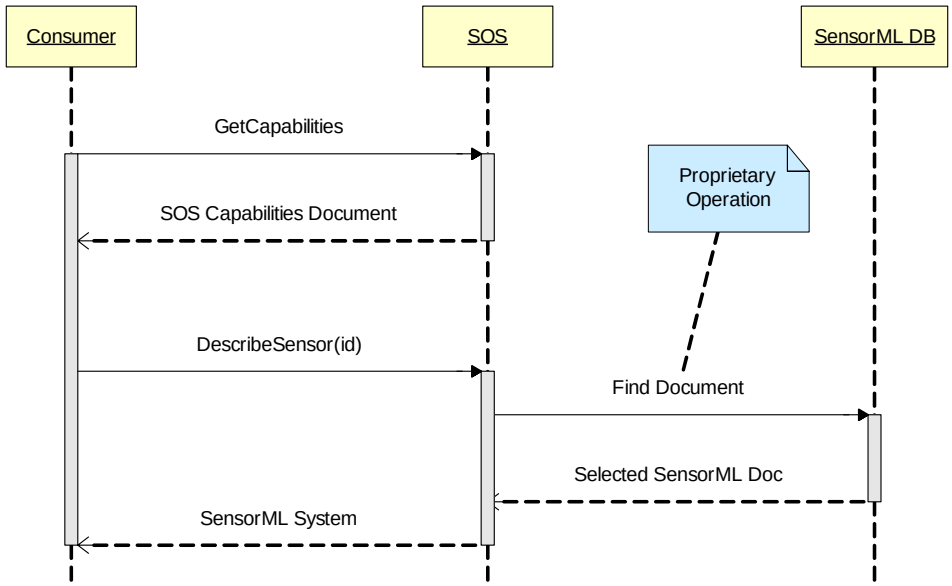


SOS GetObservation for real time streaming data

The O&M Observation still contains the description, structure and eventually encoding of the data, so that the consumer can process this information before starting to parse the incoming data stream. This means of accessing data can be used when pointing to any kind of out of band data, including raw binary and multiplexed streams (such as TML or AAF/ASF) when defined as such in the Observation.

8.5 Access to Sensor Descriptions

The SOS service also provides access to a SensorML description that defines the procedure by which observation were obtained. This procedure can be a sensor system or a simulator for instance. In order to accomplish this, the consumer of the following example first finds a sensor of interest advertised in the capabilities document provided by the service. It then issues a '*DescribeSensor'* request using the ID specified in the capabilities document. If the ID is valid, the SOS service shall then return the corresponding SensorML document after fetching it from a local database or any other proprietary system.

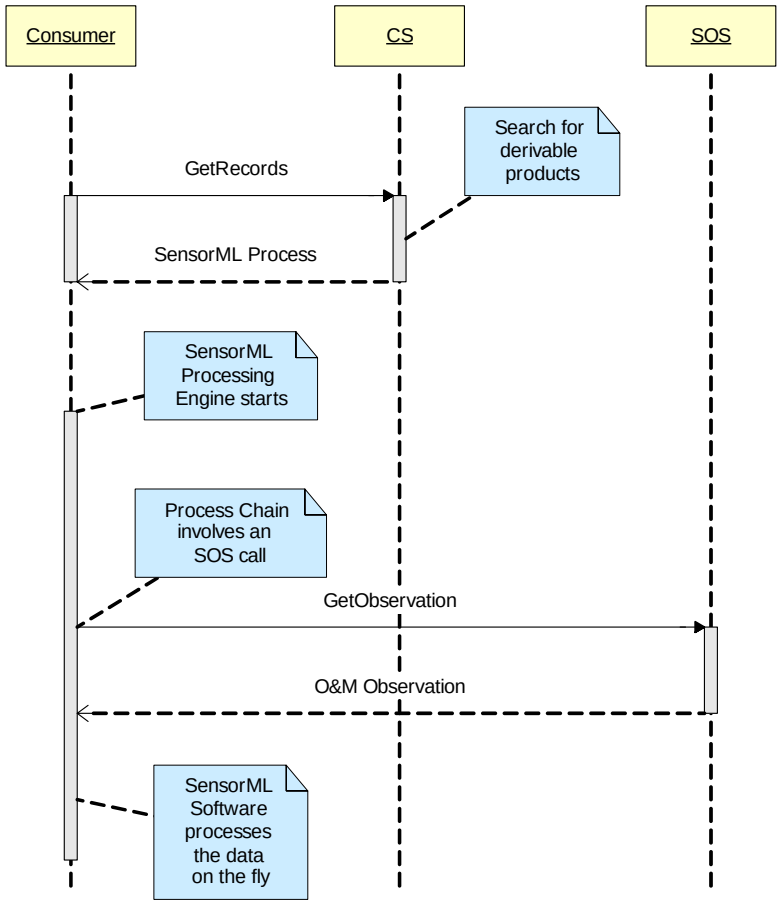


SOS DescribeSensor using SensorML documents DB

The SensorML document contains information about the sensor (or simulator) that was used to collect the observation data of the corresponding offering. This includes the calibration at the time of collection, but also the sensor location and eventually attitude. This information represents the lineage of observation data (i.e. what happened to the real phenomenon during the sampling and detecting stages of the measurement process) and is directly usable for processing of the observation data (see §6.2). It is typically used for deriving meaningful values from the raw observation data as well as geo-locating the measurements.

8.6 Data Processing using SensorML

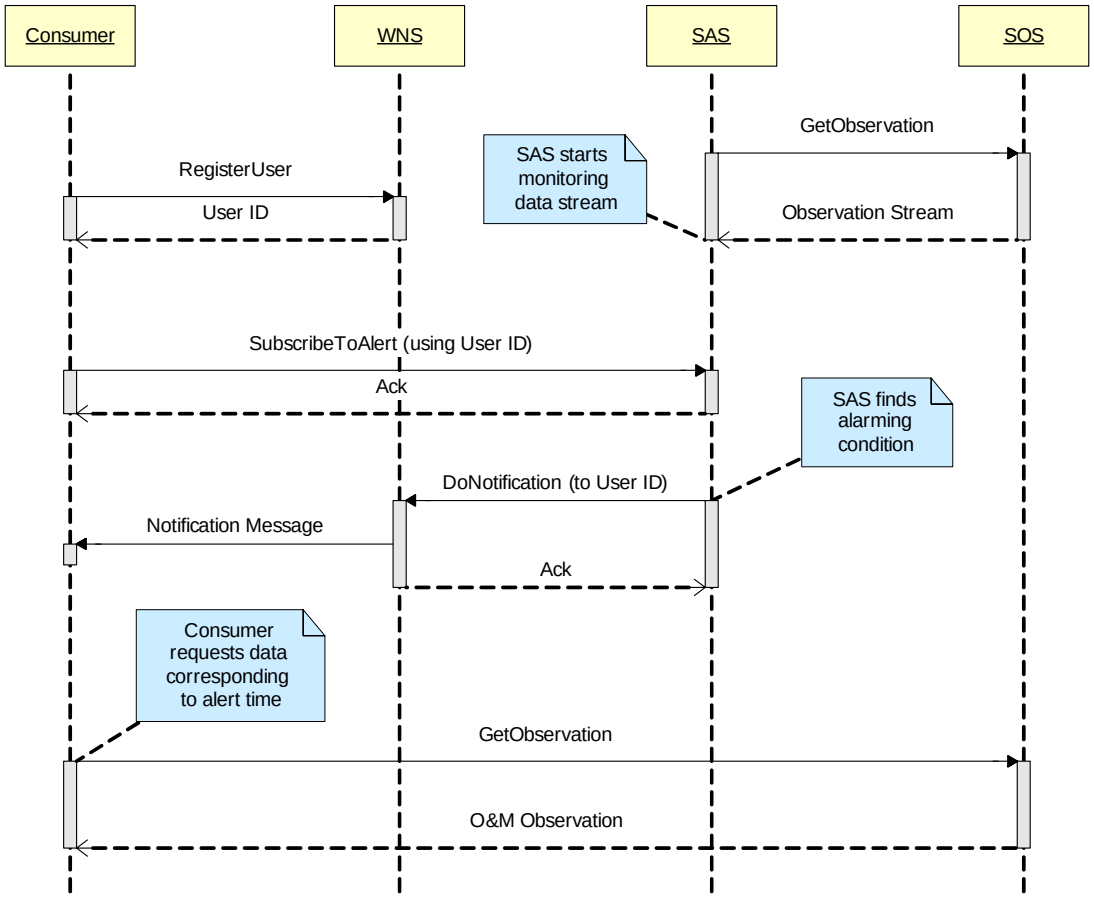
SensorML can be used very easily to process observation data coming from an SOS, and this especially if the CommonObservation model is used. A catalog service will provide the entry point to look for higher-level products, such as geolocated observations or products derived from measurements of several sensors. These derived products have traditionally been available on a server where they have been pre-computed. SensorML enables on-demand processing on the client side by providing a detailed process chain that can be applied to existing data in order to derive more useful information. The next sequence shows how a specific SensorML process can be discovered and executed by the client using generic SensorML software. The process itself can involve a call to one or more SOS in order to obtain the source data. Advantages of such a mechanism are that only exactly the desired data need to be requested and processed, and that the process can be tweaked to meet exact needs.



Processing SOS data using a SensorML Process Chain

8.7 Subscribing and Receiving Sensor Alerts

This diagram illustrates the use of three SWE services (SAS, SOS, and WNS) for subscribing to and receiving alerts generated by processing SOS data. The consumer first registers its contact information with a WNS, and gets back its user ID. The WNS is then responsible for sending future notification messages to the consumer using the provided means of communication. In the mean time a SAS service has been created and started monitoring a data stream for predefined alarming conditions. It is then assumed that the consumer somehow obtains the endpoint of this SAS service (for instance by issuing a search to a WRS). The consumer then connects to the SAS and subscribes to an alert advertised in its capabilities (‘*GetCapabilities’* request not shown for clarity), using the desired WNS address and user ID. The SAS then acknowledges that the user is subscribed and should start receiving alerts when the given condition is met.



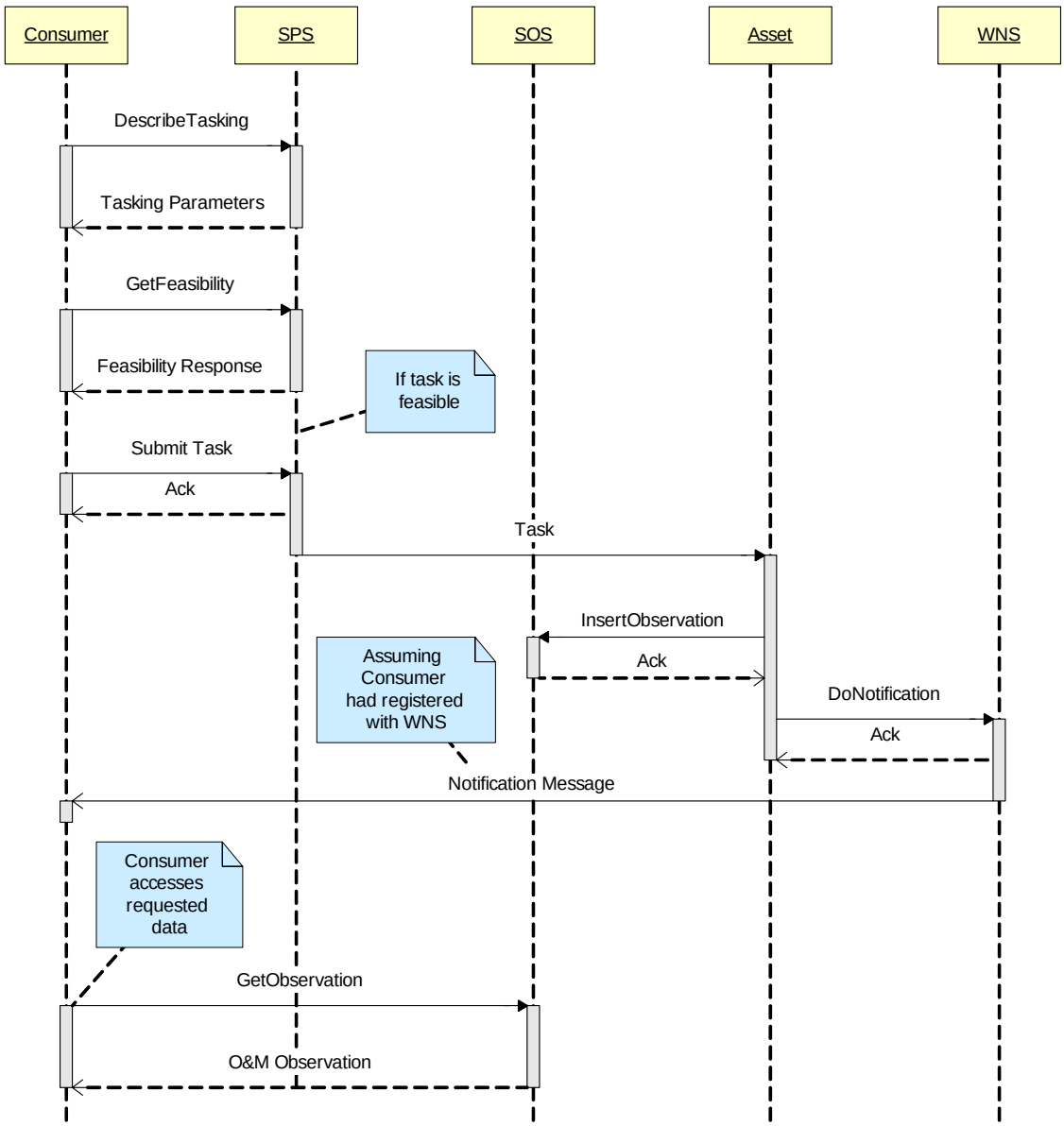
WNS - SAS - SOS collaboration for listening to alerts

When this certain condition is found in the data stream (e.g. value going over a threshold), the SAS contacts the WNS using a *DoNotification* command with the subscribed user ID and the message to be sent. The WNS then sends the notification message to the consumer using the communication means (email, phone, service connection…) specified at the time of registration.

The last part of the diagram is optional and happens only if the notification message specifies the data source (here the SOS) from which the alert was derived. This allows the consumer to contact the SOS directly and request the data corresponding to the alert. This mode of operation is especially recommended when the data used to generate the alert is bulky and thus not suited to be included within the alert message itself (The alert message should be kept as small as possible since it has to be sent to all subscribers).

8.8 Tasking Sensor Systems

This more complex diagram shows the process by which sensor tasking is made possible using the SWE framework. This sequence assumes that the consumer has already registered its contact information with the WNS and obtained a user ID (see previous).



WNS - SPS - SOS collaboration when tasking an observation collection

The consumer first issues a *DescribeTasking* request to the SPS and receives a detailed description of all taskable parameters. It then fills up desired values for these parameters and includes this in a *GetFeasibility* request in order to know if that particular task is

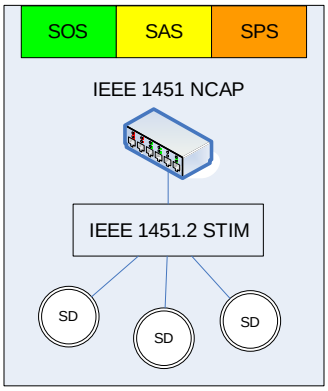
doable with this service. If the SPS's feasibility response is positive (Note that the feasibility response can also be sent asynchronously using a WNS), the consumer can submit the task for execution. This *Submit* operation specifies the task parameters as well as the WNS/UserID to be used for notification of task completion. The SPS internal software then tasks the corresponding asset to execute the mission, which is done asynchronously. While the measurement mission is executed, newly collected observations are inserted using the SOS (SOS-T) *InsertObservation* operation into an observation server. Upon completion of the task, the asset (or a third party) issues a *DoNotification* command to the WNS and the consumer is in turn alerted of the availability of the requested data. The consumer then issues a *GetObservation* on the SOS that was used to collect the data.

Real-Time low latency tasking is even possible by sending streaming commands to the SPS and receiving streaming data from the SOS, without using the notification mechanism. This scenario could be experimented during OWS-4.

9 Implementation examples

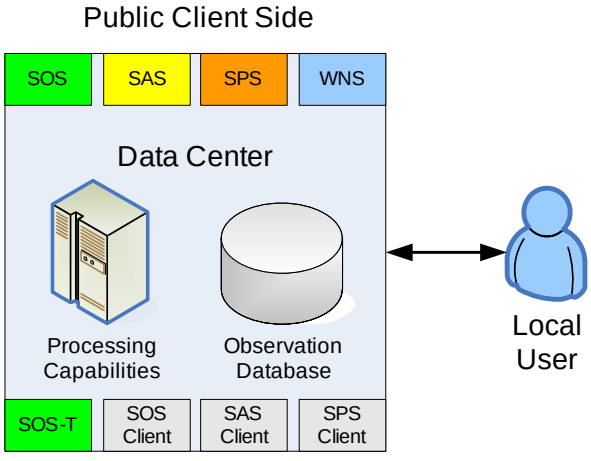
9.1 Using SWE services in a "star" network configuration

The following diagrams show the use of SOS, SAS, SPS and WNS within a specific implementation of the SWE framework inspired by Oak Ridge National Labs SensorNet. This configuration involves multiple simple nodes connected through SWE services to a central data center with much greater processing power. Sensors connected to each node are IEEE-1451 sensors that have plug n' play capability. The following diagram illustrates the configuration of each node on the network.



Node

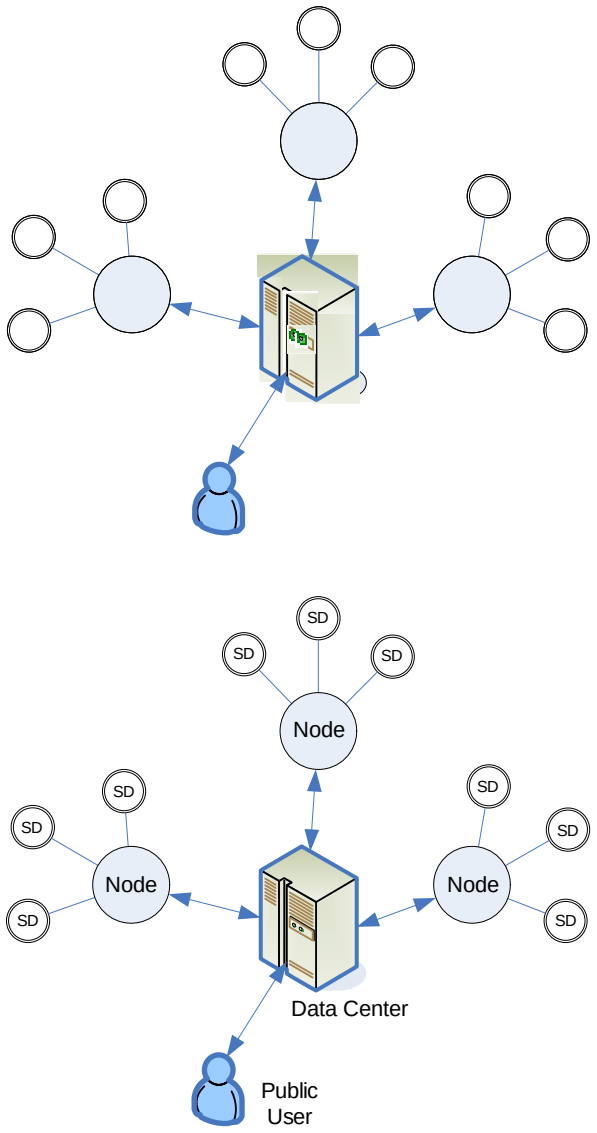
Each node provides connectivity with other network components through optional SWE interfaces. Implementations of these interfaces are expected to be very light weight since the available processing power and storage capacity on each node is very low.



In this example, each node can either incorporate a small SOS providing access to real time streaming data (if network connection is permanent) or simply inserts its observation in the data center periodically (using if network connection is intermittent, e.g. GSM).

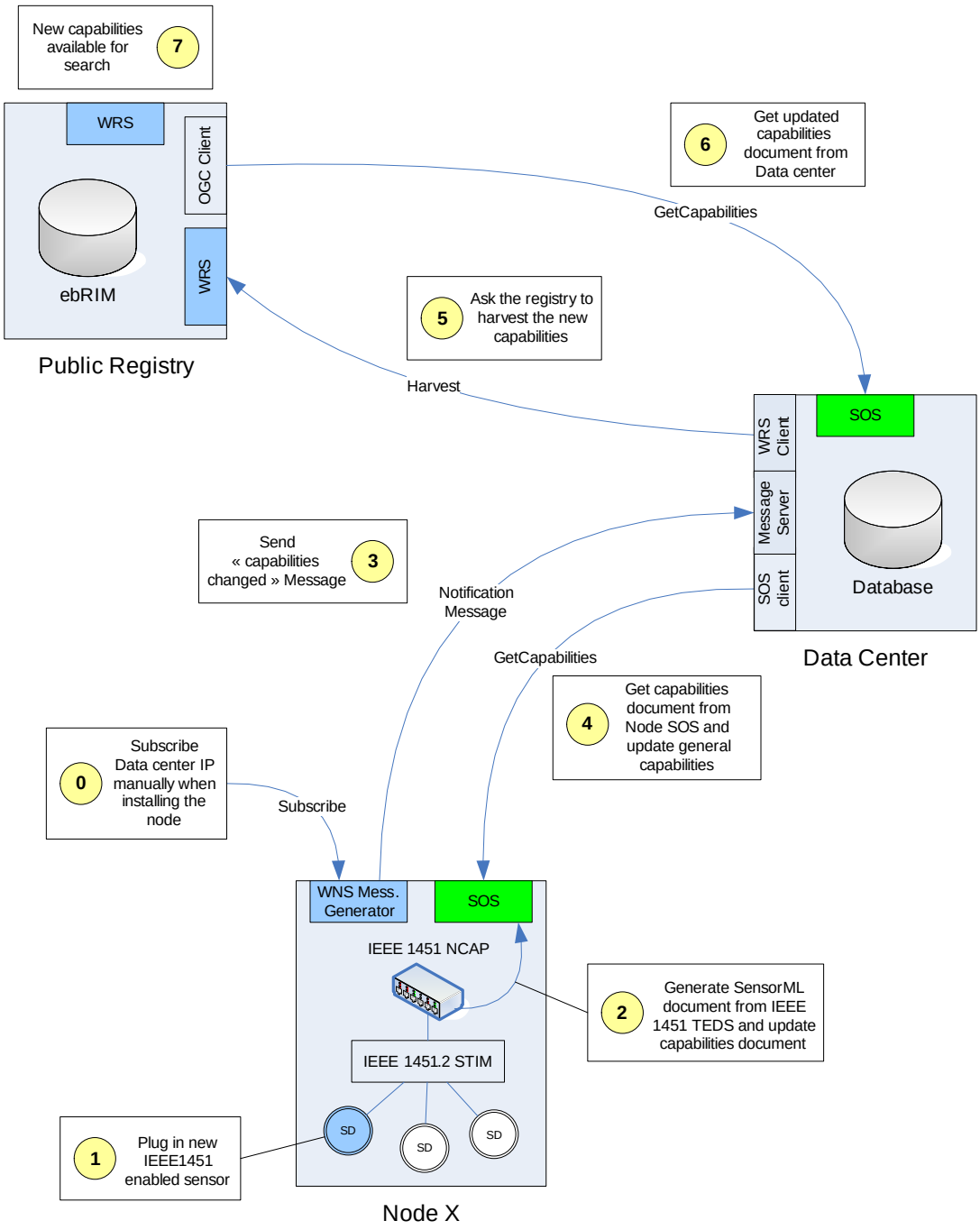
The SPS interface is only available on nodes that can achieve simple tasking operations like collecting data at a specific time or changing sensor parameters. The SAS is only available on nodes where a processing of the data is possible and used to automatically detect certain alarming conditions that will be reported to the data center. A simple WNS message generator (not the full service) is also available in order to send alert message or SPS notifications. The following diagram shows the configuration used for the data center, which is also used as a mission center (with SPS).

The data center provides many more features than a single node. It consists of a public interface accessible from the Internet and providing SOS, SAS, SPS and WNS services and a private interface only accessible from inside the sensor network and locally. The following diagram shows a global view of the network.

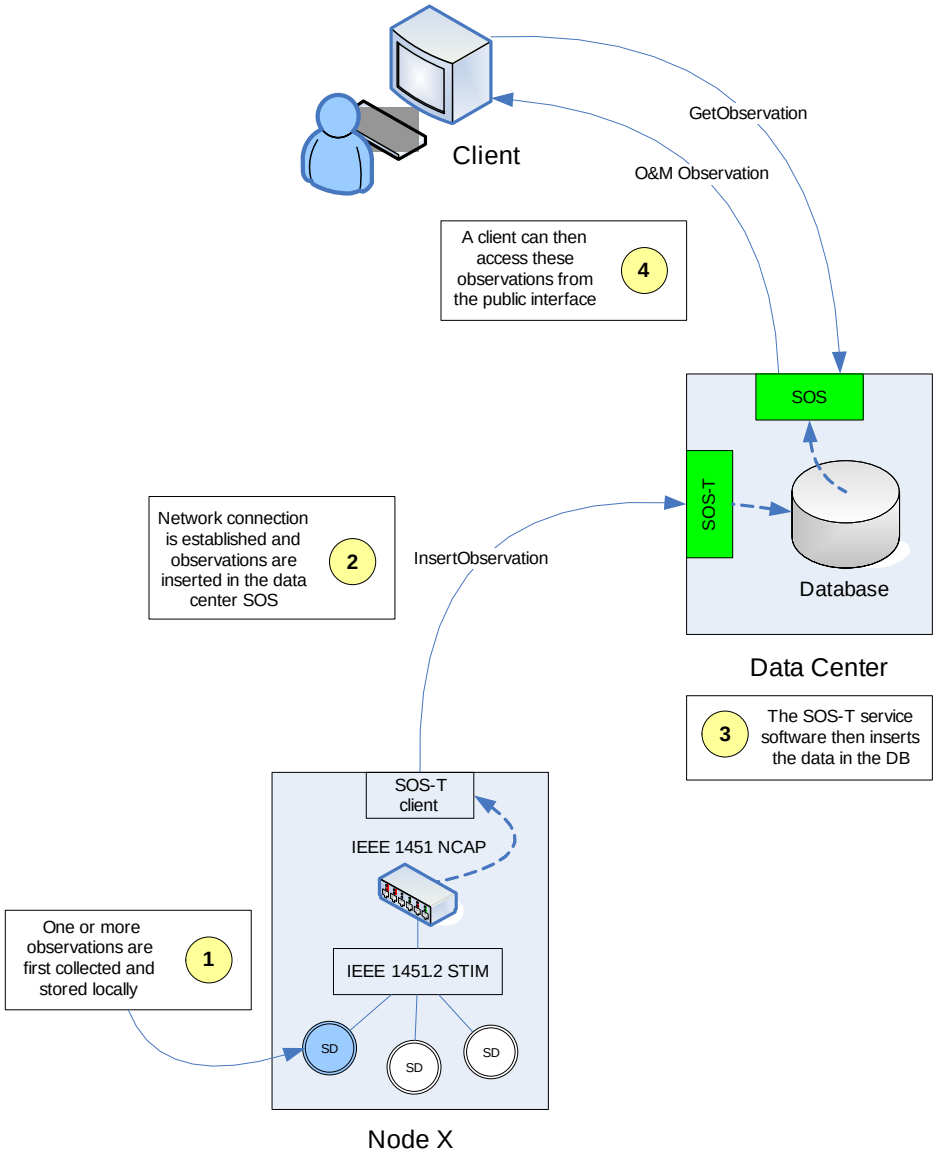


An unlimited number of nodes can of course be connected to the data center in this fashion, each of which reports data from multiple sensors. A public user doesn't have direct access to the nodes since the data center acts as a gateway. The intent with this kind of architecture is that the data center can reflect all individual nodes capabilities in one or more aggregate capabilities documents that are accessible to the public and this for all SWE services. The data center can also filter this information if some of it should not be made available to the public and eventually derive new products from raw sensor data that can be added to the global capabilities of the network.

The following diagram shows how a new sensor is added to the system in a plug n' play fashion. This shows how capabilities documents and even the public registries are automatically updated after the new sensor is inserted in one of the nodes.

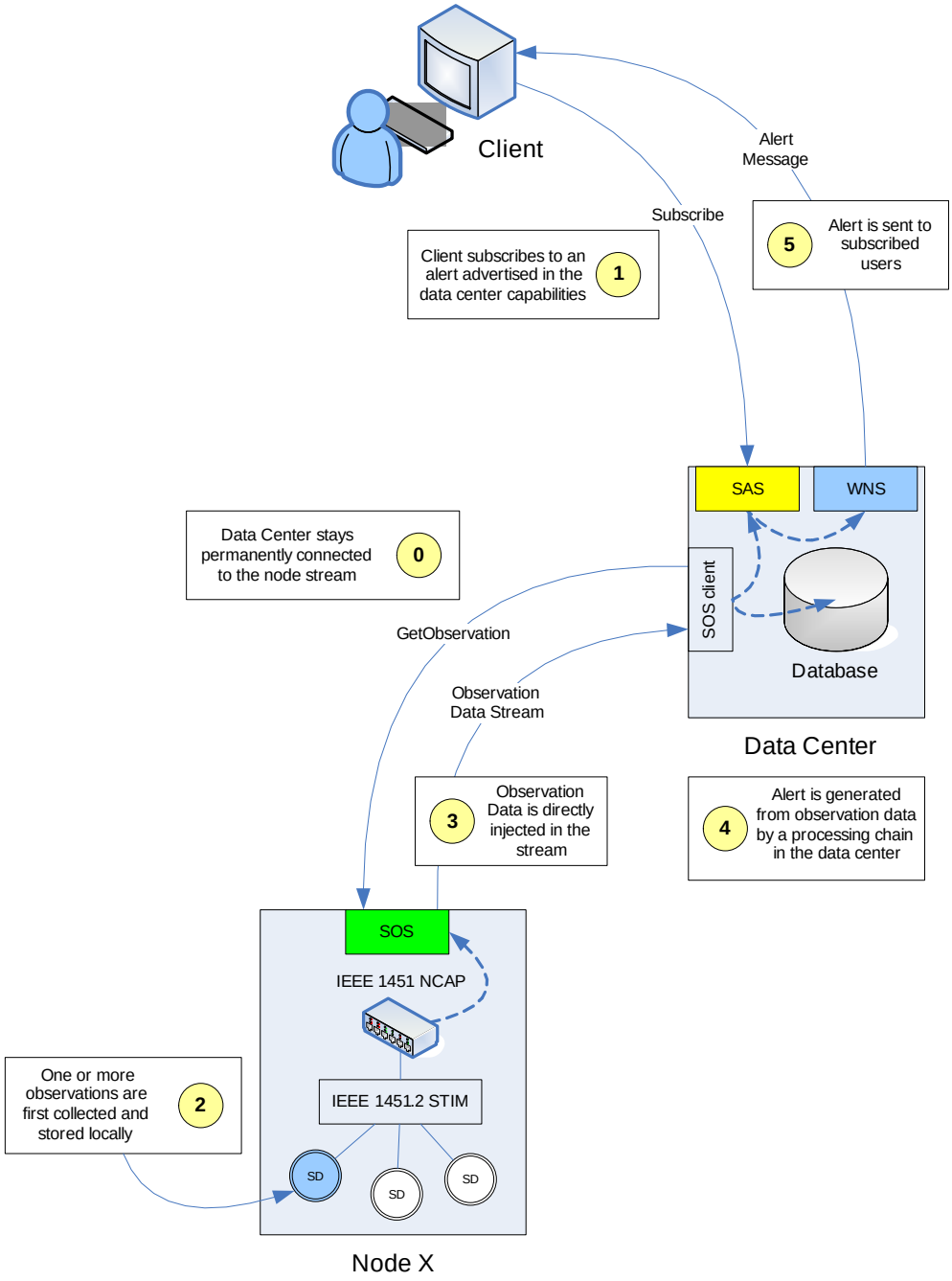


The next diagram illustrates how individual nodes can insert their observations in the main database using the data center SOS-T interface. This way of operation is very useful when node cannot keep a permanent network connection with the data center (e.g. GSM).



The new observations received at the data center can also feed a process chain used to generate advanced products, which can also be made available through the public interface. Some of these products can even be alerts generated by the data center and send to subscribers. This will be shown on the next diagram, but keep in mind that all these possibilities can be combined differently.

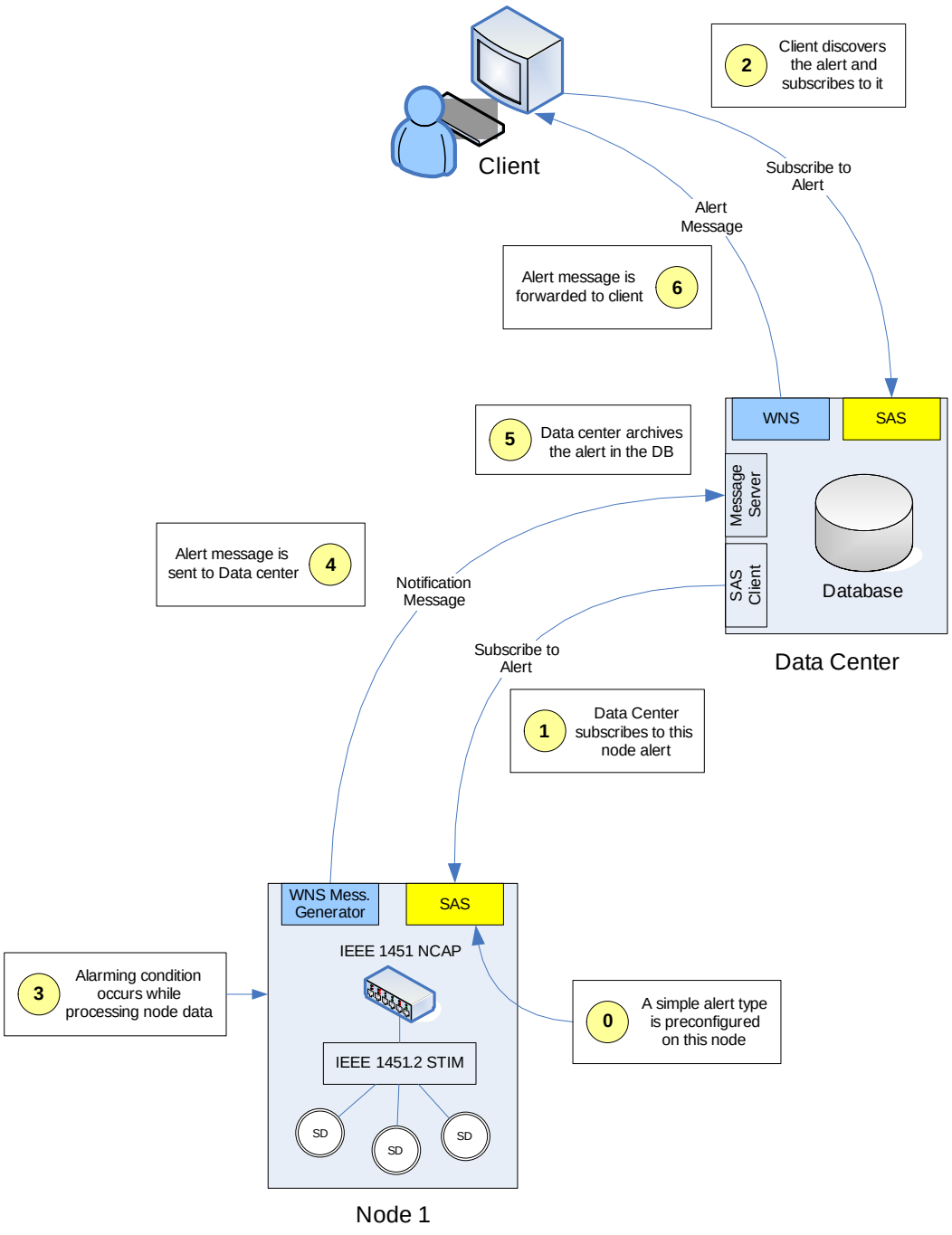
The following diagram shows the process by which nodes can stream their observation data directly to the data center when a permanent network connection is available. It also shows how an alert can be derived from these observations at the data center level and sent to all subscribed users.



This streaming configuration allows data collected at the node to be uploaded to the data center even before the collection process is terminated. This is especially interesting if observations are needed in a close to real time manner or if the node capacity is too small to store the whole data (e.g. case of video). In parallel to the alert generation process, the data center can also archive the observation data in the main database just like it was done in the previous example. This is not shown on the diagram for clarity.

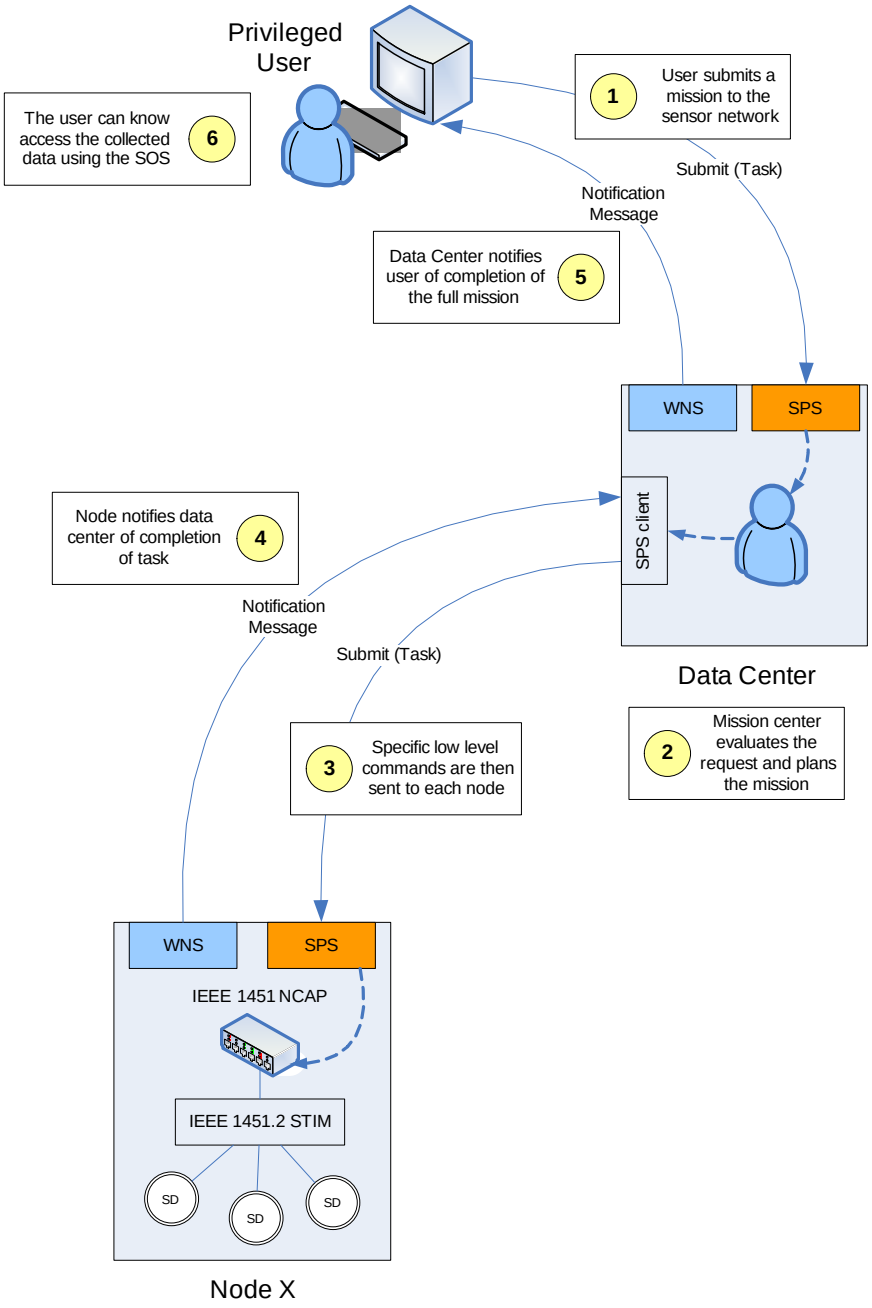
The next diagram shows how alerts can be generated at the node level and forwarded to the data center for archiving. Alerts can also be forwarded directly to the client if the later

provided contact information using the WNS and subscribed to the alert using the SAS.



Note that if the data center **always** listens to **all** alerts coming from this node, step 1 is not necessary and the notification can be issued systematically by the node software. However, adding a real SAS interface to the node provides more flexibility since different network entities will be able to subscribe to different alerts. This could be of interest if the data center uses more than one service for listening to alerts or if nodes need to listen to each other in order to cooperate on a specific measurement mission.

This last diagram shows how a hierarchy of delegated SPS can be used to issue commands to the sensor network that range from overall mission instructions at the data center level to detailed commands sent to an individual node. The task submitted at the data center is intended to be high level, whereas tasks submitted to individual nodes are low level (simple commands). This diagram doesn't show the different SPS 'discovery' operations in details (i.e. *DescribeTasking*, *GetFeasibility*, etc…) but rather assumes that these were done prior to this sequence.



10 Future Directions

An ongoing focus of the SWE architects and the SWE WG is to move all SWE component specifications to a level of maturity to warrant approval as OGC Technical Specifications. The table below shows the various levels of approval for the SWE components:

- **Sensor Model Language (SensorML)** – Best Practices (Nov 2005); Pending RFC submission (March 2006)
- **Observations and Measurements (O&M)** – Recommendation Paper; Pending Best Practices Paper and Abstract Model (March 2006)
- **Transducer Model Language (TML)** – Public Discussion Paper (Nov 2005); Pending RFC submission (March 2006)
- **Sensor Observation Service (SOS)** – Public Discussion Paper (Nov 2005); Pending RFC submission (March 2006)
- **Sensor Planning Service (SPS)** – Public Discussion Paper (Nov 2005); Pending Best Practices Paper (March 2006)
- **Sensor Alert Service (SAS)** – IPR; Pending Public Discussion Paper (March 2006)
- **Web Catalog Service: SWE Profiles** – not published

Letters of Intent have been submitted for three of the specifications, SensorML, SOS, and TML, to begin the Request for Comment stage of final approval by OGC. It is anticipated that the O&M specification will also begin the RFC process soon.

There are many areas within the SWE architecture that would benefit greatly from enhancements and better blending between components. In particular, the SWE Common components introduced during OWS3 provide a base for describing data and parameters throughout the SWE framework. During OWS3, SWE Common data types and data descriptions were incorporated to some degree within all of the SWE specifications excluding TML. However, further use of SWE Common within these specifications would strengthen the SWE architecture further by providing common models for parsing service parameters and data encodings. SWE Common datatypes must be harmonized with ISO 11404 and ISO 19103.

There is still some potential redundancy and overlap between the TML encoding and the SWE Common, SensorML, and O&M specifications. It is anticipated that further blending will occur between TML and these specification in the future.

The SensorML and TML specifications introduce the possibility of significant paradigm shifts in the implementation and application of sensors and sensor networks in the future.

Many of these paradigm shifts are already envisioned, including (1) the ability to discover, access, and process sensor observations without a priori knowledge of sensor systems, (2) the ability to package expertise and algorithms within an XML instance and to distribute and execute this algorithm anywhere on the web, (3) the ability to distribute processing of observations at any level of the sensor network, from high-powered data centers to the users hand-held computer, and (4) the ability to enable direct transmission of raw sensor observations, in real time, directly to the user with instructions on processing observations to meet that user's specific needs. Other paradigm shifts will become more apparent as more implementation of the sensor web are realized.

The SWE architects have long recognized that simulations and models should be included within the SWE design and can be supported using the existing architecture. Within the SWE framework, simulations and models are, like sensors, simply sources of observations that can be described (in SensorML), with observations can be discovered and accessed (through a SOS), with possible alerts that can be advertised and published (through a SAS), and that can be configured and tasked (though a SPS). Future work should focus on including simulations and models into the SWE framework, as well as enabling the interoperable opportunities between sensors and models.

Other directions have been recognized and are summarized in the recommendations from the SWE WG in Bonn 2005. These are listed in the section below.

10.1 OWS4 Potential SWE Directions

The following recommendations for SWE-related topics for the upcoming OWS4 Interoperability Project were submitted to the TC by the Sensor Web Enablement Working Group (SWE WG) at the Bonn TC in November 2005:

Objectives:

- Demonstrate sensor-based Web-enabled interoperable technology in support of geospatial information
- Implement, test, and refine the current SWE framework of web services and encodings within real working environments
- Extend SWE to support simulation and modeling

Recommended Directions:

- Shift major focus from design to development/implementation
    - OWS3 was perhaps 80% design, 20% implementation
    - OWS4 should perhaps be 20% design, 80% implementation and testing
- Real-time, dynamic observations
    - Non-caching of dynamic information
    - Configurable payloads and platforms
- Continue deriving/refining Catalog profiles for discovery in SWE
    - Observations, Sensors, Services
    - Terms (URNs)

- Continued harmonization within SWE components
  - TML with SWE Common, SensorML, and O&M
  - SPS, SOS, and SAS with SWE Common
- Sensor Alert Service continued development, implementation, and testing within full SWE framework
- Harmonization between SWE and other OGC efforts:
  - WFS, WNS, and SOS
  - Sensor Alert Service and Web Alert Service/WFS
  - SOS, WCS, WFS
  - WPS, SensorML, SOS
- Role-based authentication and control of services
- Inclusion and demonstration of simulations into SWE Framework
- Scaling and performance measurement and testing
- Test ACTM in SPS
- Tie into Agency/Sponsor demonstration/exercises
- Development of CITE or validation/test suites