OGC® DOCUMENT: 21-022

External identifier of this OGC® document: http://www.opengis.net/doc/PER/t17-D030

Open Geospatial Consortium

OGC TESTBED-17: SENSOR INTEGRATION FRAMEWORK ASSESSMENT ER

ENGINEERING REPORT

PUBLISHED

Submission Date: 2021-11-19
Approval Date: 2021-12-17
Publication Date: 2022-01-21
Editor: Alex Robin

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

License Agreement

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

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

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

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

Copyright notice

Copyright © 2022 Open Geospatial Consortium
To obtain additional rights of use, visit http://www.ogc.org/legal/

Note

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

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

CONTENTS

LIST OF TABLES

LIST OF FIGURES

I ABSTRACT

This OGC Testbed 17 Engineering Report (ER) documents the outcomes of a review and implementation of the Sensor Integration Framework Standards Profile (SIF-SP) v1.0.1, published by the National Center for Geospatial Intelligence Standards (NCGIS).

The Sensor Integration Framework Standard Profiles (SIF-SP) authors rightly acknowledge that sensing systems and the environments they operate in (e.g. hardware platform, computing resources, connectivity, ease of deployment, etc.) are very heterogeneous and that there will never be a single suite of technology or standards that can support the goal of providing unified access to sensor deployments employed in complex applications.

Instead, rather than trying to impose a single standard or suite of standards, the SIF-SP approach defines common conceptual models that can be mapped to existing and future standards, thus allowing integration of all these standards in a single framework.

This approach is fully compatible with the OGC Sensor Web Enablement (SWE) suite of standards that were designed for this type of integration. Thus, existing and upcoming SWE standards defined in the OGC can be used as the central pillar of a SIF implementation. The test implementation developed in this testbed, and based on OpenSensorHub, focused on demonstrating this aspect.

In addition to a thorough review of the SIF material — including standards documents, UML models and ontologies — a prototype implementation of the SIF standards was created during the Testbed using OpenSensorHub. This allowed the testbed participants to check the practical feasibility of fulfilling the SIF requirements using the OGC SWE suite of standards. Details and feedback regarding this implementation are also provided in this ER.

Suggestions to improve SIF-SP and make it an integral part of the OGC standard baseline are also provided.

NOTE: Please note that although SIF-SP was developed by military organizations, the approach and concepts are fully applicable to other domains and should thus be considered by a wider audience.

II EXECUTIVE SUMMARY

The OGC Testbed-17 Sensor Integration Thread focused on demonstrating the feasibility of implementing concepts described in the Sensor Integration Framework (SIF) standard developed by National System for Geospatial Intelligence (NSG) and United States MASINT System (USMS). A secondary objective was to demonstrate the possibility of integrating an OGC SensorThings API server with an existing SIF implementation called MASBUS.

The SIF provides a standards framework for the integration of sensor systems regardless of their technical constraints and deployment environment. The SIF thus targets systems that could

be deployed on enterprise networks as well as in Denied, Degraded, Intermittent, or Limited Bandwidth (DDIL) environments. During the Testbed, a thorough assessment of the SIF standard documents and data models was first completed. The details of this analysis are provided in this report and the recommendations from the project are as follows:

- OGC should continue to promote the SIF vision of integrated sensor systems as it is fully inline with the vision behind the OGC Sensor Web Enablement (SWE) standards.
- The SIF can be refined/developed further at OGC by defining Best Practices based on existing OGC SWE standards, not by developing new data models.
- OGC standards provide conceptual models and encodings to solve the interoperability issue at the syntactic level, but more work is needed to improve interoperability at the semantic level.
- A fully harmonized conceptual model that encompasses, not only sensor observations but also command and control aspects, needs to be developed at OGC.

The second part of this ER describes the implementation of a mediation server based on OpenSensorHub that was used to demonstrate the feasibility of heterogeneous sensor integration as envisioned by SIF. The following data sources were integrated to demonstrate the cross-domain applicability of this study:

- Unmanned Aircraft System (UAS) video and telemetry data extracted from video conforming to Motion Imagery Standards Board (MISB) standards
- UAS processed data for image and target georeferencing
- Measurement and Signature Intelligence (MASINT) sensor data obtained from a MASBUS Sensor Observation Service
- Chemical, Biological, Radiological, Nuclear, and Explosive (CBRNE) sensor data using the US Army ISA protocol
- Entire water database from the United States Geological Survey (USGS)

The mediation server is able to ingest all data listed previously and make it available via various standard interfaces, including OGC Sensor Observation Service, SensorThings API and SensorWeb API. Experiments were successfully run with both live and historical data.

There is enormous business value in solving the interoperability problem between heterogeneous sensor systems (i.e. systems of different kinds or of the same kind but from different vendors and using different proprietary interfaces). Interoperability is, as of today, the main challenge preventing the original vision of an Internet of Things (IoT) to be full realized. Indeed, the lack of standardization and interoperability between sensor vendors, system integrators and data integration/processing platforms makes it difficult for organizations dealing with large numbers of sensors systems to efficiently process data and control their assets. Although standard protocols enable communications between all kind of devices and systems at low level, large amounts of code are still required to integrate heterogeneous systems so

that they can be operated with common tools (e.g. common operating picture). Many domains such as smart cities, smart manufacturing or military operations would greatly benefit from such standardization efforts.

III KEYWORDS

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, SIF, sensor, sensor integration, SWE, MISB, ISA, API, moving features

IV PREFACE

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

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

V SECURITY CONSIDERATIONS

No security considerations have been made for this document.

VI SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Botts Innovative Research Inc.
- Sensiasoft LLC
- HeazelTech LLC
- University of Calgary

VII SUBMITTERS

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

NAME	ORGANIZATION	ROLE
Alex Robin	Botts Innovative Research / Sensiasoft	Editor
Chuck Heazel	HeazelTech / NGA	Contributor
Mahnoush Mohammadi Jahromi	University of Calgary	Contributor
Sara Saeedi	University of Calgary	Contributor

1 SCOPE

1 SCOPE

This OGC Testbed 17 (TB-17) Engineering Report (ER) represents deliverable D030 of the Sensor Integration task.

The ER presents an analysis of the Sensor Integration Framework (SIF) developed for the NSG/MASINT community and suggests a path forward for integrating the SIF vision as part of OGC's Sensor Web Enablement efforts.

The ER also contains information regarding the SIF implementation that was created during Testbed 17.

2

TERMS, DEFINITIONS AND ABBREVIATED TERMS

2 TERMS, DEFINITIONS AND ABBREVIATED TERMS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

2.1. Terms and definitions

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

2.1.1. **Sensor**

An entity capable of observing a phenomenon and returning an observed value. Type of observation procedure that provides the estimated value of an observed property at its output. [OGC 12-000]

2.1.2. **Actuator**

A type of transducer that converts a signal to some real-world action or phenomenon. [OGC 12-000]

2.1.3. System

Something of interest as a whole or as comprised of parts. Therefore a system may be referred to as an entity. A component of a system may itself be a system, in which case it may be called a subsystem. NOTE — For modelling purposes, the concept of system is understood in its general, system-theoretic sense. The term “system” can refer to an information processing system but can also be applied more generally. [ISO/IEC 10746-2:2009]

2.1.4. Location

A point or extent in space relative to a coordinate system. For point-based systems, this is typical expressed as a set of n-dimensional coordinates within the coordinate system. For bodies, this is typically expressed by relating the translation of the origin of an object’s local coordinate system with respect to the origin of an external reference coordinate system. [OGC 12-000]

2.1.5. Orientation

The rotational relationship of an object relative to a reference frame. Typically expressed by relating the rotation of an object’s local coordinate axes relative to those axes of an external reference coordinate system. [OGC 12-000]

2.1.6. Position

The location and orientation of an object relative to an external reference frame. For body-based systems (in lieu of point-based systems) is typically expressed by relating the object’s local coordinate system to an external reference coordinate system. This definition is in contrast to some definitions (e.g. ISO 19107) which equate position to location. [OGC 12-000]

2.1.7. Reference Frame (or Frame of Reference)

parameter or set of parameters that realize the position of the origin, the scale, and the orientation of a coordinate system [ISO 19111:2019]

2.2. Abbreviated terms

DDIL	Denied, Degraded, Intermittent, or Limited Bandwidth
FMV	Full Motion Video
FOI	Feature of Interest
FOV	Field of View
HTTP	Hypertext Transfer Protocol
ISA	Integrated Sensor Architecture
MASBUS	MASINT Enterprise Service Bus
MASINT	Measurement and Signature Intelligence
MISB	Motion Imagery Standards Board
MQTT	Message Queuing Telemetry Transport
O&M	Observations & Measurements
OMS	Observations, Measurements and Samples
OSH	OpenSensorHub
SensorML	Sensor Modeling Language
SIF	Sensor Integration Framework
SIF-SP	Sensor Integration Framework Standards Profile
SOS	Sensor Observation Service
SOSA	Sensor, Observation, Sample, and Actuator (Ontology)
SPS	Sensor Planning Service
SSN	Semantic Sensor Network (Ontology)
STA	SensorThings API

SWAPI	SensorWeb API
SWE	Sensor Web Enablement
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle

3

INTRODUCTION

3 INTRODUCTION

Clause 6 introduces the current SIF-SP models and architecture, as well as their mapping to OGC SWE standards.

Clause 7 presents suggested improvements to the existing SIF-SWE mappings. This includes improvements to mappings already defined in SIF as well as mappings to other OGC standards or candidate standards.

Clause 8 presents the SIF implementation developed in Testbed 17, as well as draft conceptual models to bring together the observation viewpoint provided by O&M/OMS, the system viewpoint provided by SensorML and the tasking viewpoint, and also leveraging semantics used in W3C SOSA/SSN.

Annex A provides links to JSON Schemas for SWE encodings as well as code snippets of JSON examples built using the schemas.

Annex B provides the full OpenAPI definition document and query examples for the experimental Sensor Web API that is based on the unified conceptual models discussed in Clause 8.

4

KEY FINDINGS

4 KEY FINDINGS

4.1. Applicability of existing OGC Data Models

The first key outcome of this task is the demonstration that existing OGC SWE data models (O&M, SensorML and SWE Common) go a long way to fulfil the requirements expressed in the SIF standards. For this reason, it is recommended to build on these data models and map them directly to other community standards (by defining best practices) rather than creating new data models as it was done in the original SIF documentation.

Small improvements to these existing standards, combined with the development of domain specific profiles/ontologies seem sufficient to accomplish the goal. Read the Future Work section for more detailed recommendations regarding this aspect.

4.2. Implications to OGC API Standardization

In addition to the review of SIF, a comparative review of the Sensor Observation Service (SOS), Sensor Planning Service (SPS) and SensorThings API standards was also conducted during the Testbed, leading to the following observations:

- SOS and SPS provide good cross-domain functionality, allowing access and tasking of many different sensor types, but these standards are based on older XML web services, are hard to understand and implement, and lack discovery capabilities needed for larger deployments.
- SensorThings follows a more modern REST/JSON approach that is easier to understand, but is limited to simple sensor systems as it does not have adequate support for more complex data types such as orientation and other vector quantities, imagery, video or observation profiles for example.

It seems that there is a place for a new REST/JSON OGC API (perhaps “OGC API: Sensor Systems”) that can fulfill all SIF requirements (which correspond to original SWE requirements), including support for any kind of sensor systems.

5

FUTURE WORK

5 FUTURE WORK

This section suggests a path for continuing the work initiated on SIF within the OGC Standards program. What is envisioned is a set of Best Practice documents that would provide guidelines as to how OGC web services, APIs and encodings standards (and SWE standards in particular) can be used jointly to fulfill the requirements described in the SIF reference view. Recommendations are also provided to move SIF-SP forward in the OGC standardization process so that it becomes useful for a broader community.

Some concepts in SIF and the Department of Defense Architecture Framework (DoDAF) are interesting as they help connect the dots between the Observations and Measurements (O&M) model that focuses on sensing activities and the tasking/command side of things that OGC does not have a comprehensive conceptual model for (only services and API). For example, the Performer concept as a generalization of OMS Observer is useful as many operational deployments need to deal both with observation data and command to control various assets.

In addition, the computational requirements expressed in the SIF Reference View are a good base to guide the OGC community for any further work on this topic. They are a good overview of the functionality required by organizations that need to deploy a large number of observing systems and orchestrate their use in a coherent manner.

However, the OGC, ISO and W3C have already developed conceptual models for many aspects of the problem. Therefore, defining new conceptual models as was done as part of the existing SIF work does not seem necessary. From the OGC perspective, it seems more appropriate to refine/extend these OGC conceptual models rather than defining new ones based on DoDAF concepts and mapping them to OGC models.

The following approach is thus recommended to continue this work at OGC:

- Use SIF requirements as a guide for the overall architecture (refine and extend them as required). Maintain clear mappings from requirements to solutions proposed in the models and APIs. Produce an overall architecture document explaining how OGC standards fit together to implement the requirements documented in SIF.
- Rather than DoDAF concepts, use SWE conceptual models as the core models for SIF (O&M/OMS, SensorML, SWE Common, SOSA/SSN, and see unified model presented in this ER).
- From both observation and command & control sides, work on a better conceptual model that unifies the system view and the data view (see proposal in this ER). Existing OGC conceptual models (O&M/OMS, SensorML and SWE Common) fit into this larger model.
- Map OGC SWE services to these core models in technical view 1 (SOS/SPS, SensorThings API, or a future OGC Sensors API).
- Map specialized OGC services and APIs to these core models when appropriate (e.g. EDR).

- Map any other standards to these core models through technical views (e.g. ISA, MISB, NITF, AIXM, WXXM, OPC-UA, etc.).
- Focus on defining semantics, based on W3C SSN ontology in order to provide more specific concept definitions (such as mathematical objects used to represent mechanical state, generic qualifiers for observables like spectral bands, etc.).

6

REVIEW OF SIF-SP DOCUMENTATION

6 REVIEW OF SIF-SP DOCUMENTATION

The Sensor Integration Framework (SIF) provides a standards framework for the integration of sensor systems regardless of their technical constraints. The SIF includes a Reference View that defines high level concepts and requirements and a set of Technical Views (TV). Each TV is representative of a typical deployment environment and its associated constraints. A TV defines the standards and provides guidance for the development of sensor systems within the constraints of that deployment environment.

The following three documents of the Sensor Integration Framework Standard Profiles (SIF-SP) were reviewed as part of this Testbed:

- Reference View, Version 1.0.1
- Technical View 1, Version 1.0.1
- Technical View 3, Version 1.0.1

This section provides a summary of the concepts and mappings described in these documents.

6.1. Reference View

The SIF-SP reference view is based on a subset of the DoD Architecture Framework (DoDAF) and is organized according to the principles of the Reference Model of Open Distributed Processing (RM-ODP).

RM-ODP formalism allows specifying requirements by taking different *viewpoints*, each of which allowing to address a particular set of concerns. The reference view provides information in the first 3 RM-ODP viewpoints whose definition is recalled in the following table:

Enterprise Viewpoint	Concerned with the purpose, scope and policies governing the activities of the specified system within the organization of which it is a part
Information Viewpoint	Concerned with the kinds of information handled by the system and constraints on the use and interpretation of that information
Computational Viewpoint	Concerned with the functional decomposition of the system into a set of objects that interact at interfaces

SIF also leverages some of the DoDAF concepts for which it aims at capturing information needed for the exploitation of measurement systems and their observations. These concepts are:

CONCEPT	DESCRIPTION
Resource	Data, Information, Performers, Materiel, or Personnel Types that are produced or consumed.
Activity	Work, not specific to a single organization, system or individual that transforms inputs into outputs or changes their state.
Capability	The ability to achieve a Desired Effect under specified performance standards and conditions through combinations of ways and means [activities and resources] to perform a set of activities.
Information	Information is the state of a something of interest that is materialized — in any medium or form — and communicated or received.
Performer	Any entity — human, automated, or any aggregation of human and/or automated — that performs an activity and provides a capability.

6.1.1. Enterprise Viewpoint

6.1.1.1. Use Cases

The SIF reference view defines the following use cases:

Use Case 1	Discover Sensor
Use Case 2	Describe Sensor
Use Case 3	Discover Observations
Use Case 4	Describe Observations
Use Case 5	Deliver Discrete Measures
Use Case 6	Deliver Streaming Measures
Use Case 7	Deliver Interactive Streaming Measures

Use Case 8	Set Sensor (or Actuator) Properties

6.1.1.2. Components

SIF also defines the following enterprise level components:

COMPONENT	CAPABILITY
Service Catalog	Discovery
Observation Catalog	Discovery
Sensor Manager	Command and Control
Observation Server	Delivery
Full Motion Video Server	Streaming Delivery
Imagery Server	Interactive Delivery

6.1.2. Information Viewpoint (Conceptual Models)

6.1.2.1. Resource Descriptions

SIF defines different types of resources for which robust metadata should be provided. General Resource Descriptions provide a general description suitable for any resource that can be referenced by its URI. In addition, specific resource types are defined to provide more details, an overview of which is provided in the following UML diagram and table:

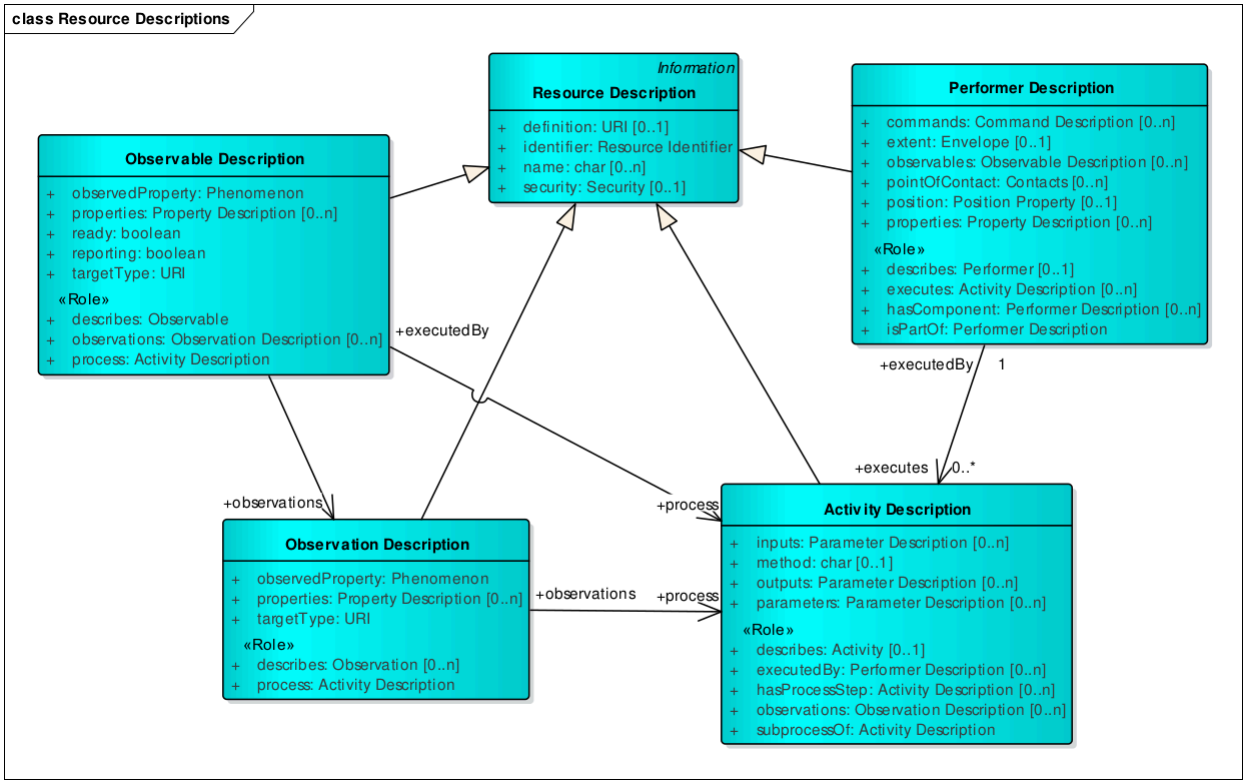

Figure 1 — SIF Resource Descriptions

Table 1 — SIF Descriptions

DESCRIPTION TYPE	DEFINITION
Performer	A Performer is an entity that performs tasks, independent of whether the entity is hardware, software, firmware or wetware. Most SIF Performers are platforms, sensors, actuators, or sensor systems.
Activity	A Performer by itself does not perform sensing. It is the Activities that are executed by the Performer that generate Observations. The SIF identifies two forms of Activities. Atomic Activities are Activities which follow a request-response pattern. Processing Activities are Activities which will execute for a period (in some cases unbounded) of time. Metadata describing the Atomic Activities are provided through the Command Descriptions. Metadata describing Processing Activities are provided through the Activity Description.
Observable	A parameter or a characteristic of a phenomenon subject to observation.
Observation	The result of measuring/observing (Activity) an observed property of a specific real-world object.

DESCRIPTION TYPE	DEFINITION
Command	Description of a command supported by a Performer
Process	An operation that takes one or more inputs and, based on a set of parameters and a methodology, generates one or more outputs.

6.1.3. Computational Viewpoint

The computational viewpoint refines SIF requirements in terms of capabilities and activities available from the various components of the architecture. The following table summarizes requirements listed for each group of capability:

Table 2 — SIF Messaging Capabilities

CAPABILITY	DESCRIPTION
Direct	Entity sends a message to a specified recipient or group of recipients.
Pub/Sub	Entity broadcasts a message which can be received by any number of recipients who subscribes to it.
Message Oriented Middleware	Messaging system where the routing decisions are made by the messaging infrastructure, not the sender or recipient.

Table 3 — SIF Discovery Capabilities

CAPABILITY	DESCRIPTION
Resource Discovery	Discover resources regardless of their type.
Content Discovery	Discover information resources, including data sets, files, and entries in relational databases.
Process Discovery	Discover activities that are available for execution.
Performer Discovery	Discover performers available to the user as well as their full description.

Table 4 — SIF Delivery Capabilities

CAPABILITY	DESCRIPTION
Discrete Delivery	Discrete delivery is the delivery of an information resource, either by providing the entire resource itself or a reference to it.

CAPABILITY	DESCRIPTION
Streaming Delivery	With streaming delivery, the client is provided a service access point from which the resource is available as a continuous stream of data (e.g. full motion video, audio, other real-time data feeds).
Interactive Delivery	Interactive delivery allows the user to select which parts of a (large) resource to deliver at a time (e.g. large satellite imagery products).
Register	Submit a resource to a delivery service so that it will be available for access.

Table 5 — SIF Command Capabilities

CAPABILITY	DESCRIPTION
Command	Issue a command that is used to start, modify, or stop the execution of a process/activity.
Actuation	Issue a command that is used to start, modify, or stop the execution of a process/activity.

Table 6 — SIF Sensing Capabilities

CAPABILITY	DESCRIPTION
Command and Control	Covers all of the control and status activities needed to place the sensor where it needs to be, establish the parameters for a successful collection, start and stop the collection.
Sensor Collection	Covers the activities performed by emitters, collectors, and all of the support processing required to produce a usable information resource.

Table 7 — SIF Human-Computer Interface (HCI) Capabilities

CAPABILITY	DESCRIPTION
Query Builder	An interface to help building complex queries that are used to filter query results and pub/sub data.
Content Presentation	Tools to transform and display the raw data in a way that a human can comprehend.
User Input	A way to convert a human input to a machine readable format understood by other components.
Response Handler	A piece of software to receive and handle messages from pub/sub data sources asynchronously.

Table 8 — SIF Information Assurance Capabilities

CAPABILITY	DESCRIPTION
Identification	The process of discovering the true identity (i.e., origin, initial history) of a person or item from the entire collection of similar persons or items.
Authentication	The act of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system.
Authorization	The process of defining and maintaining the allowed actions.
Access Control	The ability to allow or deny access to applications and resources at a granular level, such as per-user, per-group, and per-resources.
Confidentiality	The ability to protect data so that unauthorized parties cannot view the data.
Integrity	The ability to assure that data has not been altered in an unauthorized manner. Data integrity covers data in storage, during processing, and while in transit.
Non-Repudiation	The ability to provide protection against an individual falsely denying having performed a particular action.

6.2. Technical View 1

The SIF-SP Technical View 1 provides mapping rules between the SIF Reference View and the following OGC Sensor Web Enablement standards:

- SensorML
- Observations & Measurement
- Sensor Observation Service
- Sensor Planning Service

Please read the Clause 7 section for more details and discussion about these mappings.

6.3. Technical View 3

The SIF-SP Technical View 3 provide general mapping rules with Tactical Denied, Degraded, Intermittent, or Limited Bandwidth (DDIL) environments, including a specific test suite for implementations based on the Integrated Sensor Architecture (ISA) DoD standard.

Please read the Clause 8 section to learn more about how the ISA and MISB DoD standards were mapped to SWE conceptual models.

7

MAPPINGS TO SWE STANDARDS

7 MAPPINGS TO SWE STANDARDS

This section discusses suggested improvements to the SIF mappings to better take into account existing and future OGC standards.

For existing mappings, suggested changes are highlighted in blue in the following tables.

7.1. SensorML

SensorML is a robust language that is well suited for describing the following types of resources and concepts:

SIF/DoDAF	Person, Activity, Platform, Process
	Procedure, ObservingProcedure, Observer, Host
SSN	System, Platform, Sensor, Actuator

7.1.1. Mappings with SIF Descriptions

Table 9 — Mapping between SIF and SensorML Classes

COMMENTS
Observation
Observable Property
Both type level (i.e. description of a sensor model) and instance level (i.e. description of a particular instance/ deployment of a sensor model) descriptions can be implemented with SensorML. The link between the two is done using the typeOf property
Simple Process, Aggregate Process,

COMMENTS
When the feature of interest is the system itself (e.g. the aircraft/platform that we measure location of), a SensorML class can be used to model it. In this case the system can be both a Performer and a Real-World Object

Table 10 — Mapping between Activity and SensorML Process

SENSORML ABSTRACTPROCESS
[illegible]
0..1 outputs n
0..1 parameters n
Performer can be referenced using a specific contact role or classifier
0..1 characteristics, processing capabilities, Information documentation
Depending on the desired level of granularity, a SimpleProcess or an AggregateProcess can be used. When an AggregateProcess is used, components and connections are used to define the steps. When a SimpleProcess (atomic process) is used, the steps/algorithms can be described in the method section, either textually or using a more formal language such as MathML.

Table 11 — Mapping between Performer and SensorML Process

SENSORML PHYSICALCOMPONENT
[illegible]

SENSORML PHYSICALCOMPONENT
Recommend using gml:id for local identification only, not to store any representation of the unique identifer which should aim at being globally unique
0..n name, identifiers Main name is provided by the name property; Further application specific identifiers can also be provided
The main definition specifies what this physical component is through a URI which resolves to an element in the SIF-SP ontology. More (domain) specific classifiers can also be provided (e.g. sensor or platform type).
0..1 boundedBy extent The boundedBy property could be used although in this case it is meant to be the extent within the performer operates, not the extent of the feature itself
0..n contact, pointOfContact Modeled as ISO 19115 CI_ResponsibleParty
In SensorML, position can be specified inside the component or system itself, or in the description of the system that encapsulates it (hence the 0..n cardinality).
0..n characteristics, capabilities, properties, parameters
Commands are modeled using parameters marked as ‘updatable’
state
When a PhysicalSystem is used, components can be described individually
0..1 attached To Used to navigate up the hierarchy of components
Performer is associated with Activity/Process though inheritance

7.2. SWE Common Data Model

SWE Common is a robust low-level data model that is useful for providing detailed metadata about the structure and semantics of data records as well as the data record values themselves in different possible encodings (CSV, XML, JSON, binary).

In particular, SWE Common can be used to implement both schema and instance level information for the following concepts:

Observation properties, Performer parameters, SWE / DoDAF Command inputs/outputs/parameters, Activity inputs/outputs/parameters
Observation result, Observation parameters
Result, SystemCapability, SystemProperty, SSN / SOSA ActuatableProperty, Condition

Work has been done on the JSON encoding for both SWE Common Data Model v2.0 and SensorML v2.0/2.1 standards.

See JSON schemas provided in Annex A.

7.2.1. Supported Encodings

Various encodings are supported by the SWE Common standard. Compression methods are also supported via the BinaryEncoding (See Clause 8.13 in the implementation section).

7.3. Observations & Measurements (O&M & OMS)

“Observations and Measurements” (O&M) version 2.0 has been published both as an OGC and as an ISO Standard. Version 3.0 is under development and is now called “Observations, Measurements and Samples” (OMS). OMS will also be released as a joint OGC/ISO standard.

The OMS v3 conceptual models introduce important refinements to the O&M models, including among others:

- The new **ObservationCollection** class;
- A clear distinction between an **Observer** and the **ObservingProcedure** it performs;
- The *proxy feature of interest* and *ultimate feature of interest* concepts that generalize O&M's concepts of *sampling feature* vs. *sampled feature*.

Although the O&M/OMS design seems to be driven mostly by communities using in-situ sensors, the more flexible design of OMS v3 will help solve many problems related to encoding inefficiencies encountered with O&M v2.

A UML diagram of the core OMS v3 conceptual models is provided below:

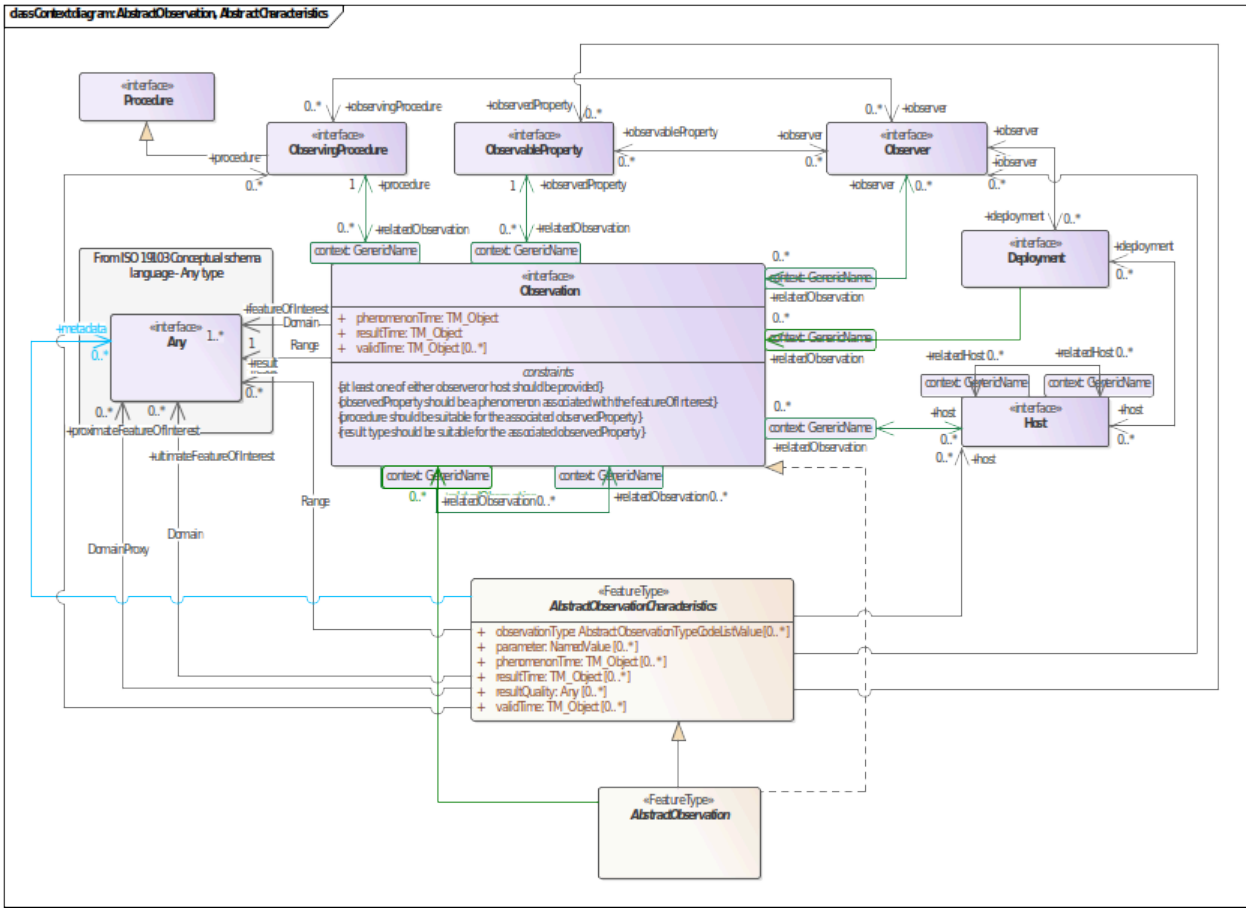

Figure 2 — OMS Conceptual Observation Model

7.3.1. Definitions

Before presenting a mapping between O&M/OMS and SIF, the key concepts of O&M/OMS are summarized below:

7.3.1.1. Observation

An **Observation** is an act associated with a discrete time instant or period through which a number, term, or other symbol is assigned to a phenomenon. An observation involves application of a specified **Procedure**, such as a sensor, instrument, algorithm, or process chain. The procedure may be applied in-situ, remotely, or ex-situ with respect to the sampling location. The result of an observation is an estimate of the value of a property of some feature.

7.3.1.2. ObservableProperty

An observable quality (property, characteristic) of the feature of interest that may be observed.

7.3.1.3. FeatureOfInterest

The Feature that is the subject of the **Observation**. More discussion on modeling features of interest is provided in section Clause 8.10.

7.3.1.4. Procedure

A description of steps performed. More specifically, an **ObservingProcedure** describes the steps performed by an **Observer** to generate an **Observation**. Likewise, a **SamplingProcedure** describes the steps performed by a **Sampler** in order to extract a **Sample** from a sampled feature.

7.3.1.5. Observer (OMS v3 only)

An identifiable entity that may generate **Observations** by performing a certain **Procedure**.

7.3.1.6. Host (OMS v3 only)

A grouping of **Observers** for a specific reason.

7.3.1.7. Deployment (OMS v3 only)

Information on the assignment of an **Observer** to a **Host**.

7.3.1.8. Observation Result

The result of an **Observation**. The O&M conceptual models are agnostic of the result type and different result types can be modeled to accommodate the needs of different domains.

However, one thing to note is that OGC SWE Common Data Model (see Clause 7.2) provides a generic model for describing result types across communities. It allows defining both simple and more complex result types by providing a semantically robust schema and encoded values for the observation result.

7.3.2. Mappings with SIF Concepts

The following table provides recommended mappings between O&M v2 classes and SIF / DoDAF concepts:

Table 12 — Mapping between SIF and O&M v2 Concepts

COMMENTS
Observation
Observable Observable Property
Both Performer and Activity are mapped to Procedure which caused confusion in many applications of O&M in the past. These two concepts have been separated in OMS v3 for this reason (see below).
Activity Procedure

The following table provide recommended mappings between OMS v3 classes and SIF / DoDAF concepts:

Table 13 — Mapping between SIF and OMS v3 Concepts

COMMENTS
Observation
Observable Observable Property
Procedure, Observing Activity, Sampling Procedure

NOTE: SIF defines a “realizes” relationship between Observation and Observable. We think that may not be the best way to model this relationship as the Observation result is an “estimate” of the value of the Observable at a given point it time.

7.3.3. Encodings

XML encodings for O&M 2.0 have been released as part of OGC O&M [10-025r1]. They are used by the Sensor Observation Service in particular.

JSON encodings for O&M 2.0 have been released as an OGC Discussion Paper (OGC 15-100r1) and can be used as a basis for OGC APIs serving observation resources.

7.4. Sensor Web Enablement (SWE) Services (SOS & SPS)

In this section, mappings between SIF and both Sensor Observation Service (SOS) and Sensor Planning Service (SPS) are considered. Both of these services are XML based web services and have been OGC standards since 2007 and 2010 respectively.

SOS and SPS information models are almost entirely based on O&M and SensorML/ SWECommon, so the mappings defined above are still valid in the context of these services.

7.4.1. Mappings with SIF Information Model

SIF CLASS	SWE CLASS	COMMENTS
Performer	SensorML process or system description in SOS/SPS Describe Sensor and SOS InsertSensor	
Activity	-	
Observation	O&M Observation in SOS GetObservation and Insert Observation	
Observation Description	ObservationOffering in SOS Capabilities	
Observable	via reference to external ontologies only	
Command Description	SWE Common parameters within SPS DescribeTasking Response	
Command	SWE Common encoded message	
Real-World Object	GML Feature in GetFeatureOfInterest	

7.4.2. Mappings with SIF Enterprise Viewpoint

SIF USE CASE	SWE SERVICE OPERATION	COMMENTS
Discover Sensor	SOS/SPS GetCapabilities	Need to browse through offerings
Describe Sensor	SOS/SPS DescribeSensor	
Discover Observations	SOS GetCapabilities	Need to browse through observation offerings

SIF USE CASE	SWE SERVICE OPERATION	COMMENTS
Describe Observations	SOS GetResultTemplate	Does not include all static observation metadata, just the result structure and observables
Deliver Discrete Measures (Server to Client)	SOS GetObservation or GetResult	Filtering can be used
Deliver Discrete Measures (Sensor to Server)	SOS InsertObservation or InsertResult	
Deliver Streaming Measures	Non-standard websocket extension available in OSH SOS implementation	Filtering can be used
Set Sensor (or Actuator) Properties	SPS Submit	
Deliver Interactive Streaming Measures	*Out of Scope*	

7.4.3. Mappings with SIF Computational Viewpoint

COMMENTS
SOS-SPS Get Direct operations Messaging via HTTP GET.
SOS-SPS Deliver Get Message operations

COMMENTS
via HTTP GET.
Publish
WS-Notification Notify Notify operation

7.5. SensorThings API

SensorThings API (STA) v1.1 is an API based on REST principles and OData. It provides access to both observations and tasking capabilities. The UML diagram of the core SensorThings classes/ entities is provided below:

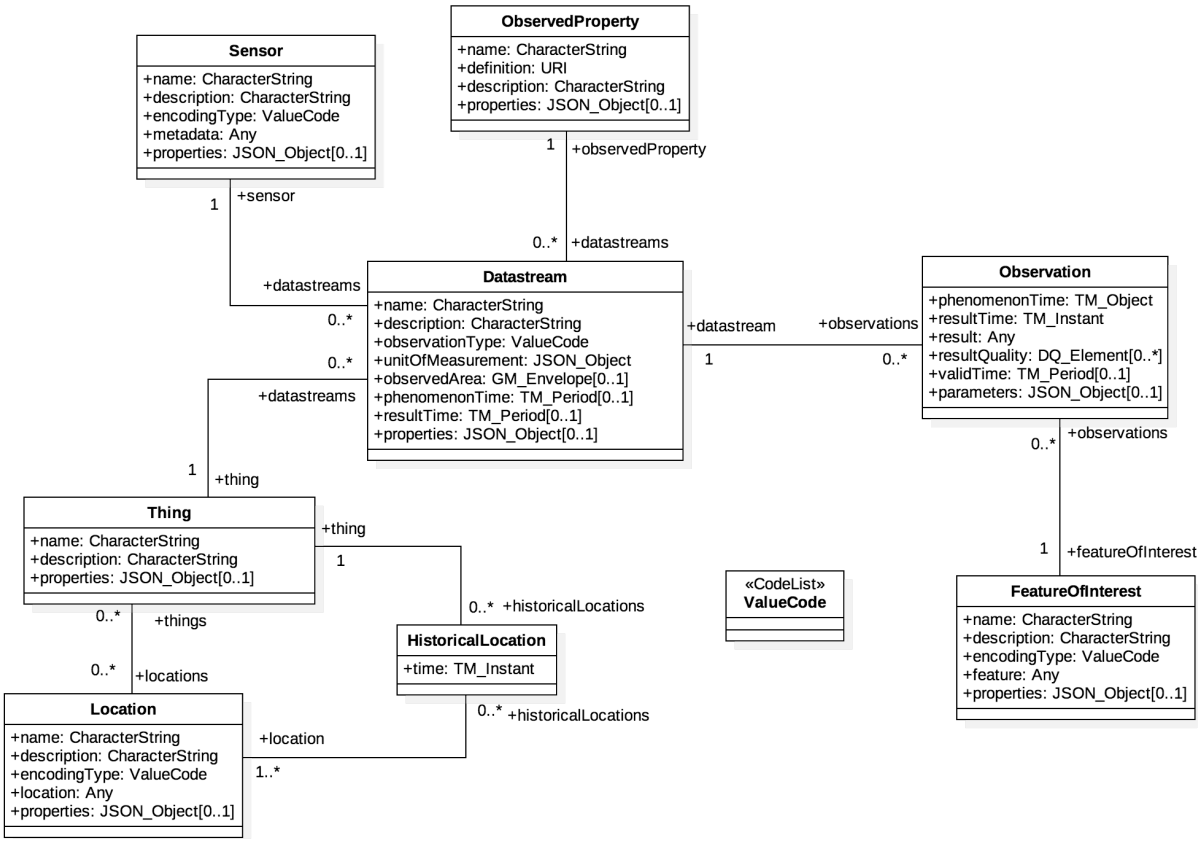

Figure 3 — SensorThings Core Entities

The additional classes/entities used for tasking are illustrated below:

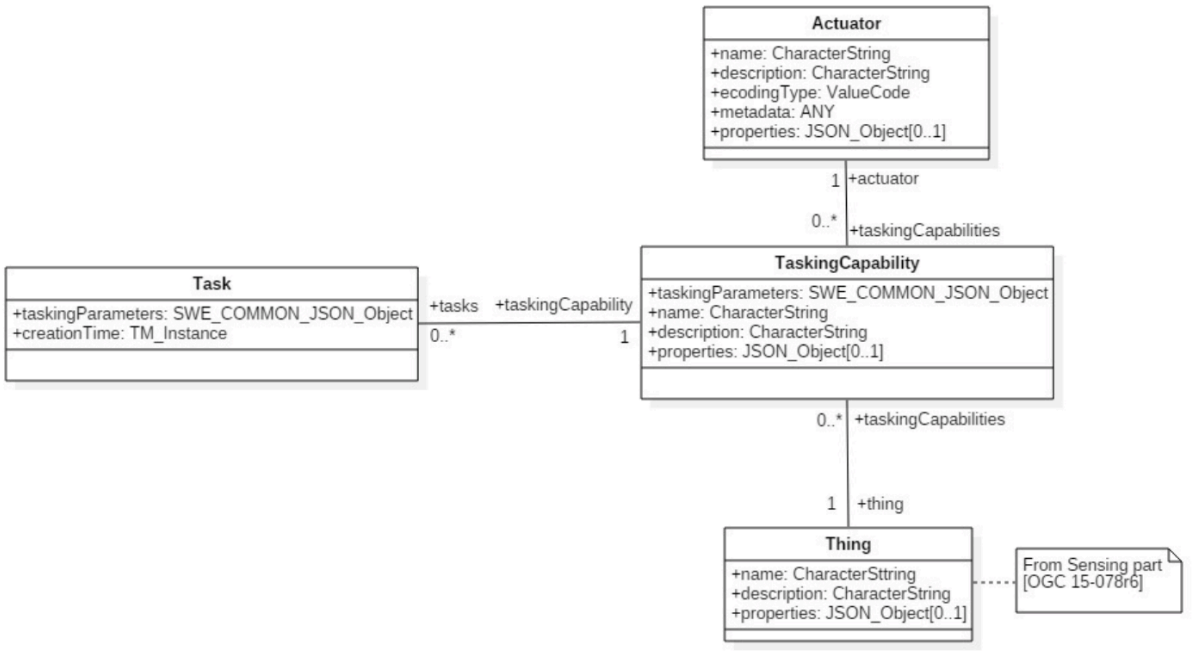

Figure 4 — SensorThings Tasking Entities

7.5.1. Mappings with SIF Information Model

COMMENTS
Aggregate performers such as systems composed of several sensors can be modeled as Things. Only very limited metadata (corresponding to SIF Resource Description) is defined by the standard. More could be added via profiles.
Observation
Limited metadata defined by the standard. More could be added via profiles.
Observed Property
Task

7.5.2. Mappings with SIF Enterprise Viewpoint

SIF USE CASE	SENSORTHINGS CAPABILITY	COMMENTS
Discover Sensor	Browse/search through Thing and Sensor entities	
Describe Sensor	Fetch details of a particular Sensor	Not much metadata provided on Sensor entities
Discover Observations	Browse/search through ObservedProperty and Datastream entities	
Describe Observations	Fetch details of a particular Datastream	
Deliver Discrete Measures	Fetch Observations from Observation collections (HTTP GET)	Filtering can be used
Deliver Streaming Measures	Subscribe to Observations from one-or-more Datastreams (MQTT)	Filtering can be used
Set Sensor (or Actuator) Properties	Post Task to a TaskingCapability (HTTP POST), Publish Task to a TaskingCapability (MQTT)	
Deliver Interactive Streaming Measures	-	

7.5.3. Mappings with SIF Computational Viewpoint

SIF CAPABILITY	SENSORTHINGS OPERATION/PROTOCOL	COMMENTS
Direct Messaging	HTTP & MQTT	
Post Message	HTTP POST/PUT	
Deliver Message	HTTP GET	
Publish-Subscribe Messaging	via MQTT extension	
Publish	MQTT publish from client to server	
Subscribe	MQTT subscribe	
Notify	MQTT publish between server and client	

SIF CAPABILITY	SENSORTHINGS OPERATION/PROTOCOL	COMMENTS
Route Message	-	

7.6. SensorWeb API (Draft)

The SensorWeb API is a REST API under development by the OpenSensorHub team. This API targets a full implementation of the SWE vision (i.e. observation access, streaming, tasking, processing and discovery using O&M, SWE Common and SensorML) using a REST approach.

The SensorWeb API uses the HTTP/REST protocol for discovery and historical data access, as well as Websocket and MQTT bindings providing streaming and publish/subscribe capabilities for both observation and command data feeds.

7.6.1. Mappings with SIF Information Model

COMMENTS
Additional semantic allows specifying the type of performer (e.g. system, sensor, actuator, platform, etc.).
Modeled as a SensorML SimpleProcess or AggregateProcess
Modeled from OMS v3 ObservationCollection.
Description sub-resource provides the detailed data structure of the observation results using SWE Common.
Properties can be used both as “observable properties” and “actuatable/changeable properties”
Similarly to Datastream, this includes the command schema defined as SWE Common.

7.6.2. Mappings with SIF Enterprise Viewpoint

SIF USE CASE	SENSORWEB API CAPABILITY	COMMENTS
Discover Sensor	Browse/search through System and Procedure resources	
Describe Sensor	Fetch details of a particular System (SensorML)	
Discover Observations	Browse/search through ObservableProperty and Datastream resources	
Describe Observations	Fetch details of a particular Datastream and its schema sub-resource	
Deliver Discrete Measures	Fetch observations from Observation collections (HTTP GET)	Filtering can be used
Deliver Streaming Measures	Stream real-time Observations from a Datastream (Websocket), Subscribe to Observations from one-or-more Datastreams (MQTT)	Filtering can be used
Set Sensor (or Actuator) Properties	Post Command to a CommandStream (HTTP POST), Stream Command to a CommandStream (Websocket), Publish Command to a CommandStream (MQTT)	
Deliver Interactive Streaming Measures	-	

7.6.3. Mappings with SIF Computational Viewpoint

SIF CAPABILITY	SENSORWEB API OPERATION/PROTOCOL	COMMENTS
Direct Messaging	HTTP & MQTT	
Post Message	HTTP POST/PUT, Websocket	
Deliver Message	HTTP GET, Websocket	
Publish-Subscribe Messaging	via MQTT extension	
Publish	MQTT publish from client to server	
Subscribe	MQTT subscribe	

SIF CAPABILITY	SENSORWEB API OPERATION/PROTOCOL	COMMENTS
Notify	MQTT publish between server and client	
Route Message	-	

8

TEST IMPLEMENTATION

8 TEST IMPLEMENTATION

This section presents the implementation based on OpenSensorHub (OSH) that was developed during Testbed-17 to demonstrate the feasibility of SIF concepts.

8.1. Scope

The scope of the test implementation is to demonstrate the feasibility of integrating various standard interfaces as well as legacy systems into a common framework by following principles of the SIF-SP. The implementation focuses on the use of existing and upcoming OGC standards to implement the SIF/SWE vision. That is, to allow interaction with heterogeneous sensor systems in a unified manner, including access to system metadata, observation data and ability to send commands.

The implementation focuses on demonstrating the following aspects in particular:

- The extent of the metadata that is possible to transmit using OGC standards;
- The integration with existing legacy systems without requiring changes or duplication of existing datastores;
- The ability to transmit a wide range of data types, from simple in-situ measurement to high data-rate video feeds.

The following diagram illustrates the data sources that were successfully integrated during the Testbed:

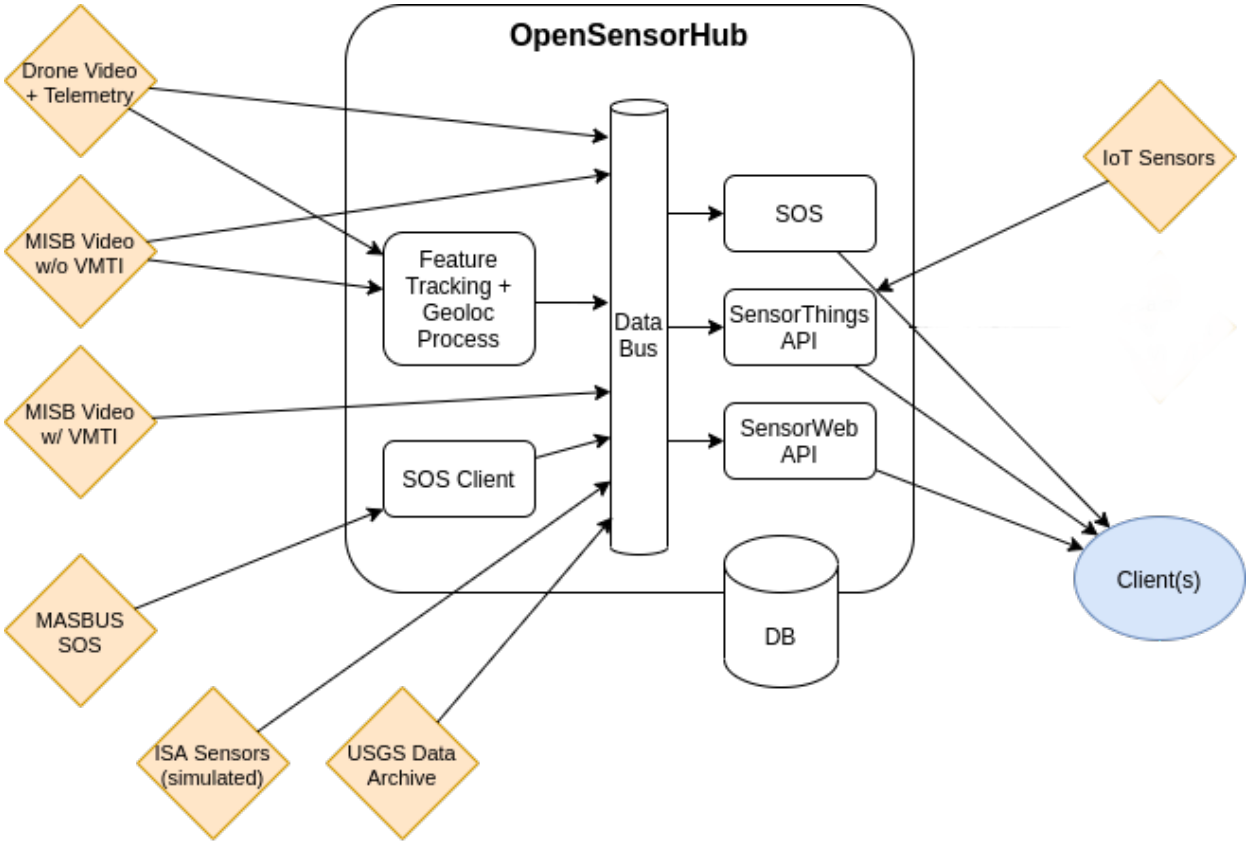

Figure 5 — OSH Based SIF Implementation

8.2. OpenSensorHub

OpenSensorHub (OSH) is a Java framework that can enable any sensor, actuator, process, forecast model, robot, or system to be discovered, accessed, and controlled through OGC/SWE standard services and APIs.

OSH can also act as a bridge between protocols, either between SWE protocols themselves (e.g. SensorThings-SOS bridge) or between SWE and other proprietary protocols or community standards (e.g. OnVIF video cams to SWE bridge). OSH achieves this goal by defining Java APIs based on the SWE conceptual models, towards which sensor drivers, web services/APIs, and datastore connectors can be developed.

A simplified view of OSH architecture is shown on the diagram below:

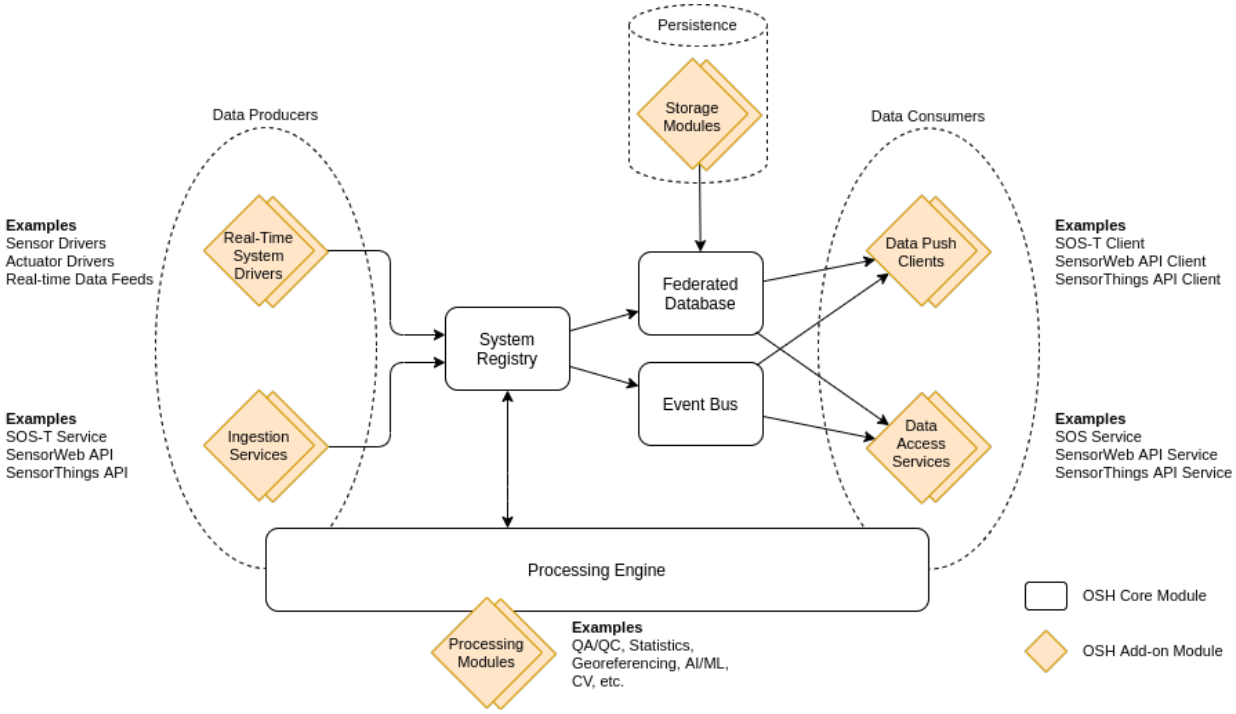

Figure 6 — OSH Architecture

8.3. Core Models

OpenSensorHub is built on conceptual models defined in O&M/OMS, SensorML and SWE Common. Additional semantics are provided by the OGC/W3C SSN ontology as well as the SensorML ontology from Botts Innovative Research that is browsable at http://sensorml.com/ont.

In addition to leveraging concepts defined in the above standards, OSH is based on a unified implementation model that connects together classes from O&M/OMS, SensorML, SWE Common, and SSN. The model is presented here as it could be used as the base for further work on SIF.

Below is a simplified UML diagram showing the main classes of the model. The diagram also illustrates how the model "realizes" classes of the main standard conceptual models it is based on (shown with a dashed outline):

- The `om` prefix denotes classes from the O&M and OMS model.
- The `ssn` prefix denotes classes from the SOSA/SSN ontology.
- The `sml` prefix denotes classes from the SensorML model.
- The `iso` prefix denotes classes from the ISO feature model.

The role of many OSH extensions (e.g. sensor drivers, datastore connectors, etc.) is to adapt legacy data models and formats to this internal unified model.

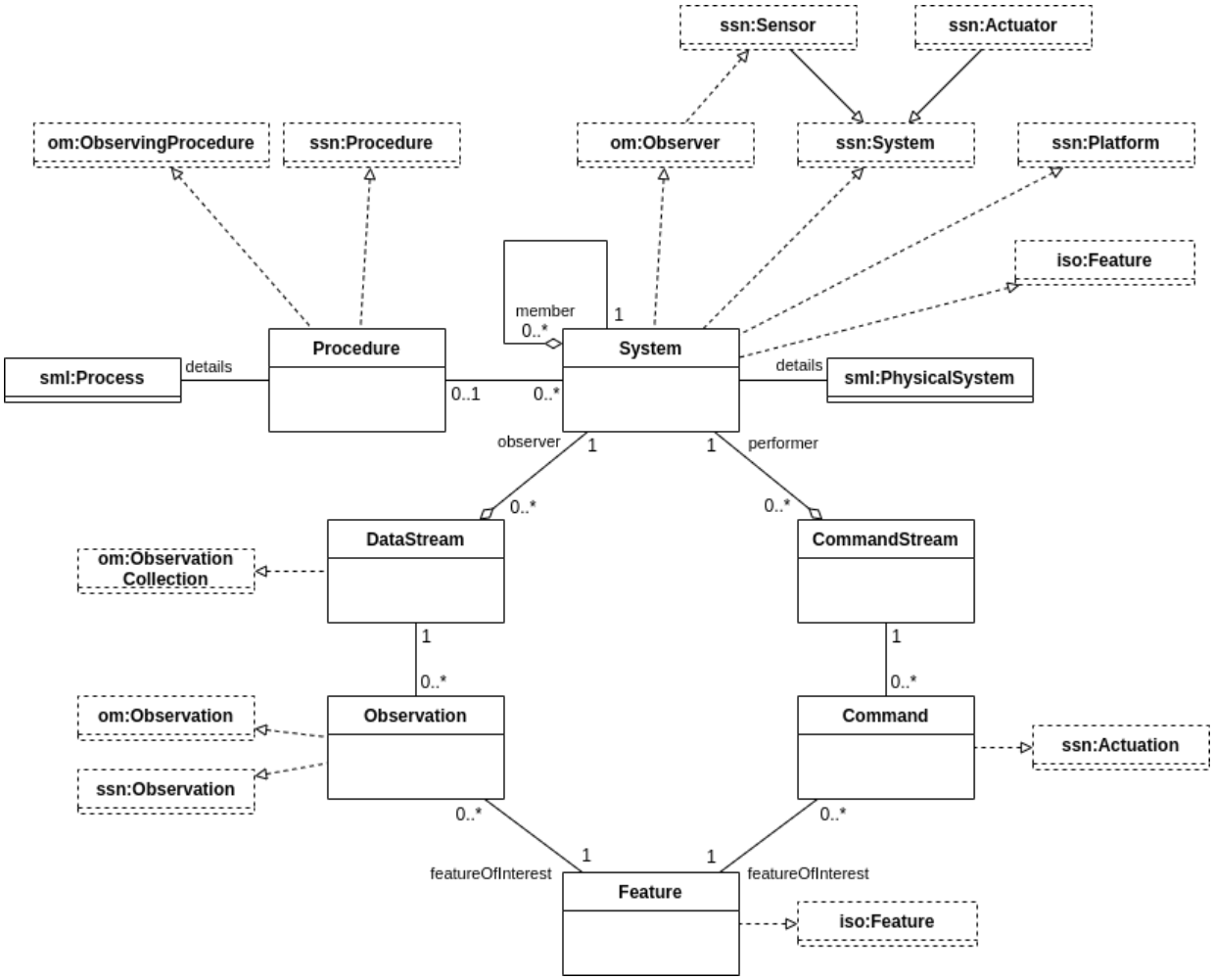

Figure 7 — Unified SWE Model

In this model, the following definitions hold:

a) As in SSN, a System represents an instance of an observing system that implements a particular Procedure. A System can be a physical piece of hardware, a logical process executed in a processing unit or even a human performing predefined steps in a lab or in the field. The difference between these types of systems is made by semantic tagging rather than creating different classes.

b) As in SSN, a System can have members (i.e. subsystems) that are themselves Systems. This allows describing more complex hierarchical systems with the desired level of granularity.

c) Contrary to SSN, an OSH System can also be used to model a Platform, which can then have its own Datastreams and command channels.

d) Any System can have any number of Datastreams. Therefore describing the complete hierarchy of System components is not necessary, even when providing all data generated by these components. If desired, the System can be modeled

as “black box” (i.e. without sub components) with multiple datastreams (although technically the data comes from individual system components). The same is true for commands.

e) Both Observation results and Command parameters are described and encoded using the SWE Common standard, which allows efficient binary encoding. See Clause 8.13 for details about the supported encoding and codecs.

8.4. Multi-protocol Bridge

OSH implements the idea of a protocol bridge envisioned by SIF. Thus ingesting data using one protocol and expose it with another is possible. OSH takes care of the proper transformation and tries to minimize the loss of information in the process.

For example, the following transformations have been demonstrated in the Testbed:

- Ingest data from a standard SOS server and expose through SensorThings API and SensorWeb API.
- Ingest data using SOS-T and expose through SensorThings API and SensorWeb API.
- Ingest data using SensorThings API and expose via SOS and SensorWeb API.
- Ingest MISB data/metadata and expose via SOS, SensorThings API and SensorWeb API.
- Ingest ISA protocol buffer data and expose via SOS, SensorThings API and SensorWeb API.
- Ingest data from legacy United States Geological Survey (USGS) webservice and expose via SOS, SensorThings API and SensorWeb API.

Internally, all OSH components are interconnected using the datastore and event bus APIs that make this data interchange possible. Thanks to this architecture, interoperability between protocols is possible for both real-time and historical cases, and for both observation and command data flows.

8.5. Integration Points

OpenSensorHub can integrate with a legacy system in different ways:

a) By developing a “Sensor Driver” (or System Driver) in a JVM programming language (Java, Groovy, Scala, Kotlin, etc.) that implements the OSH sensor

API. Such a driver supports direct access to any locally connected device (e.g. via serial, USB, Bluetooth) or digital data feed (e.g. a datastream flowing from a remote server such as a message queue). A driver is designed to capture real-time data coming to/from a system and convert between legacy formats and OGC SWE data models. The driver can handle both observation and command flows and is also responsible for collecting metadata about the system it connects to.

b) Another way of integrating an external device or other real-time data feed is by developing a piece of software that runs separately from OSH and exchanges data with OSH using one of the available transactional APIs (SOS-T, SensorThings API and SensorWeb API). Note that since this extension will run outside of the OSH node itself, it can be developed in any programming language. This method can also be used to ingest large volumes of historical data in batch mode.

c) Finally, integrating directly with an existing datastore containing historical observations and/or metadata is also possible using the Datastore API. In this case, OSH acts as a proxy to an underlying datastore with or without maintaining its own cache (e.g. LRU cache of most commonly accessed data). This type of integration is especially of interest for large data collections that are difficult or costly to duplicate.

The following integrations have been implemented or improved during Testbed 17:

8.6. Sensor Drivers/Adapters

Sensor drivers (or System drivers) provide translators from/to lower level protocols such as the ones described in SIF-SP Technical View 3 (e.g. ISA). In this Testbed, the following drivers have been developed and showcased:

8.6.1. ISA Protocol Driver

This driver is used to connect systems that implement the ISA protocol, and has two modes of operation:

a) Connection to real-time ISA sensor feed supporting the protobuf encoding;

b) Generation of metadata and observation data for a simulated ISA sensor network.

In both cases, the driver exposes system level metadata as well as descriptions for its datastreams and command channels. A single instance of the driver is able to decode data for a large number of ISA compliant systems, each of which is able to generate observations and receive commands.

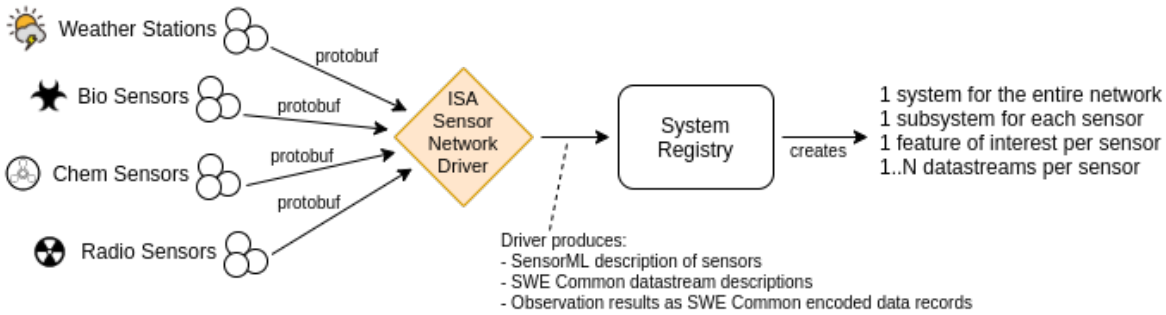

Figure 8 — ISA Driver Diagram

In addition to observable properties and observation values, sensor identifiers, classifiers, characteristics and capabilities are obtained from ISA encoded data and converted to SensorML. The following sensor properties are supported:

- Manufacturer name
- Model number
- Software version
- Sensor type
- Operating Range / Voltage
- Operating Range / Current
- Operating Range / Battery capacity
- Measurement resolution
- Measurement accuracy
- Integration time
- Sampling frequency

A snippet of the generated SensorML metadata in the JSON encoding is provided in Annex A.2.1. Note that properties are semantically tagged using concept URIs that are resolvable to external ontologies.

The following sensors & datastreams are available through this driver:

SYSTEM TYPE	DATASTREAMS	COMMENTS
Radiation Sensor	Radiological Reading, Radio Link Status (Link State), Sensor Location	RADIO001 to RADIO004 in the demonstration, with simulated triggers when radioactive source is nearby
Biological Sensor	Biological Reading, Radio Link Status (Link State),	BIO001 to BIO004 in the demonstration, with simulated random measurements

SYSTEM TYPE	DATASTREAMS	COMMENTS
	Sensor Location	
Chemical Sensor	Chemical Reading, Radio Link Status (Link State), Sensor Location	CHEM001 to CHEM002 in the demonstration, with simulated random measurements
Weather Sensor	Atmospheric Humidity, Atmospheric Precipitation, Atmospheric Pressure, Atmospheric Temperature, Atmospheric Wind, Electrical Status, Radio Link Status (Link State), Sensor Location	ATM001 to ATM002 in the demonstration, with simulated random measurements

8.6.2. MISB UAS Driver

This driver is used for ingesting a live or pre-recorded MPEG-TS video data stream with embedded MISB.0601 KLV tags (UAS Metadata Set). This type of datastream is typically generated by STANAG compliant Unmanned Aircraft System (UAS).

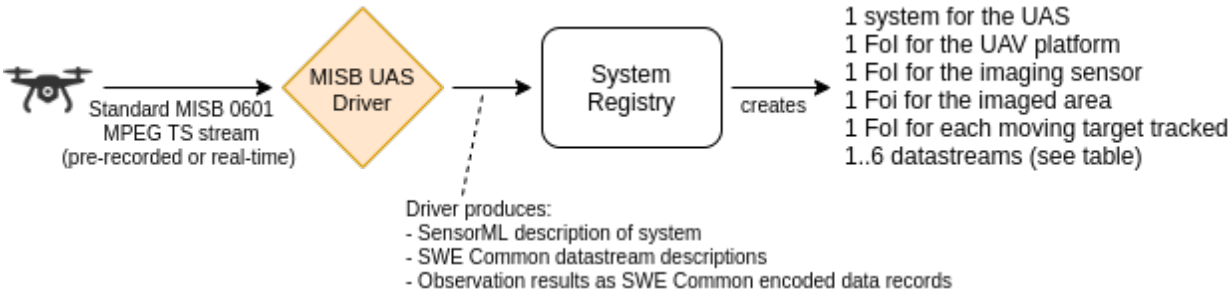

Figure 9 — MISB Driver Diagram

The following datastreams are made available by this driver:

DATASTREAM	FEATURE OF INTEREST	COMMENTS
Platform Orientation	UAV Platform	Relative to local NED reference frame
Sensor Location	UAV Imaging Sensor	Geographic location (EPSG 9705)
Sensor Orientation	UAV Imaging Sensor	Relative to UAV platform
Sensor Parameters	UAV Imaging Sensor	Variable Field of View (FOV)
Georeferenced Image Frame	Imaged Area	Geographic coordinates (EPSG 4979)

DATASTREAM	FEATURE OF INTEREST	COMMENTS
Video Stream	Imaged Area	H264 frames wrapped in SWE Common binary stream
Video Moving Targets (VMTI)	Moving Targets	Target locations in both image and geographic reference frames

This driver showcases important capabilities provided by OGC SWE standards:

a) Be able to identify and describe the feature of interest for each datastream.

b) Be able to describe the relative positioning of various components of a remote sensing system.

Point 1 is of utmost importance in this UAS use case because a single MISB system generates datastreams that provide observations about different real-world objects, including:

- The drone platform
- The imaging sensor
- The remotely sensed area
- The moving targets detected and tracked in the video

Clearly identifying these real-world features is essential for a correct interpretation of the data (e.g. the location of the sensor or the location of the remote target on the ground are two very different things). The “feature of interest” (FoI) concept of O&M is used here to provide metadata about these objects and correctly associate them with datastreams and individual observations.

Point 2 is equally important for proper use of the positioning information provided by the system. The SWE Common Vector construct is used to unambiguously describe the frames of reference with respect to which location and orientation parameters are expressed (see Clause 8.11 for details). There are at least four main frames of reference in this use case:

- 3D geodetic coordinate reference systems. Note here the difference between EPSG 4979 corresponding to WGS84 lat/lon coordinates + height above ellipsoid and EPSG 9705 corresponding to WGS84 lat/lon coordinates + height above mean see level (MSL).
- The local horizontal North-East-Down (NED) reference frame.
- The platform reference frame that is rigidly attached to the aircraft/UAV.
- The sensor reference frame that is rigidly attached to the camera sensor (the sensor is typically mounted on a gimbal and can thus be rotated with respect to the platform).

The full SWE Common descriptions for MISB datastreams are provided in Annex A.2.2. Note that properties are semantically tagged using concept URIs that are resolvable to external ontologies.

8.7. Datastore Connectors

Another way data was integrated during Testbed 17 is by proxying existing data stores using OSH datastore API. Two datastores were integrated:

- The MASBUS SOS server, using the standard OGC SOS protocol.
- The USGS Water Database, using the legacy webservice protocol and RDF text format.

8.7.1. MASBUS SOS Server

This connector is used to connect to a MASBUS DSOSRI service hosted by Riverside Research using the standard OGC SOS protocol. The connector connects regularly to the MASBUS SOS server in order to retrieve all new sensors' metadata records and observations, and updates/ publishes the info to the OSH integration hub.

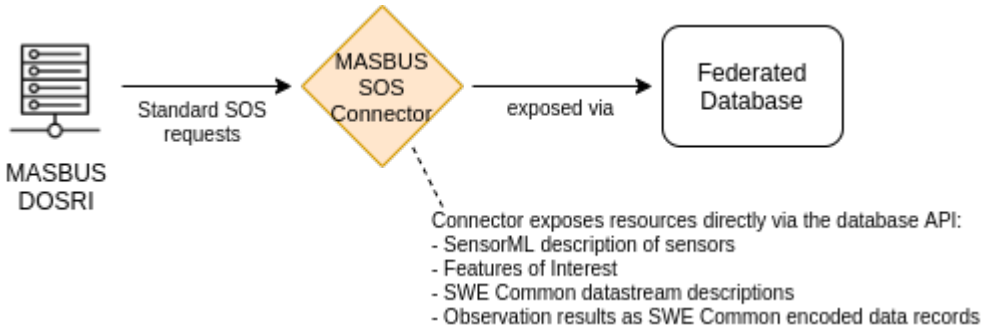

Figure 10 — MASBUS Connector Diagram

The following sensors & datastreams are made available by this connector:

SYSTEM TYPE	DATASTREAMS	COMMENTS
Omnisense Detector	Passive Infrared, RGB Radiance, Vibration	Omnisense-10712, Omnisense-10721, Omnisense-10728,
TRSS Sensor	TARACT Reading = Detection of objects of interest	Sensors with IDs TRSS:MM*`

8.7.2. Legacy USGS Datastore

This connector is used to proxy the entire USGS water database made available through USGS Instantaneous Values Web Service. This database contains historical data for millions of observation sites and hundreds of observed properties. Information about monitoring sites is fetched using the USGS Site Web Service.

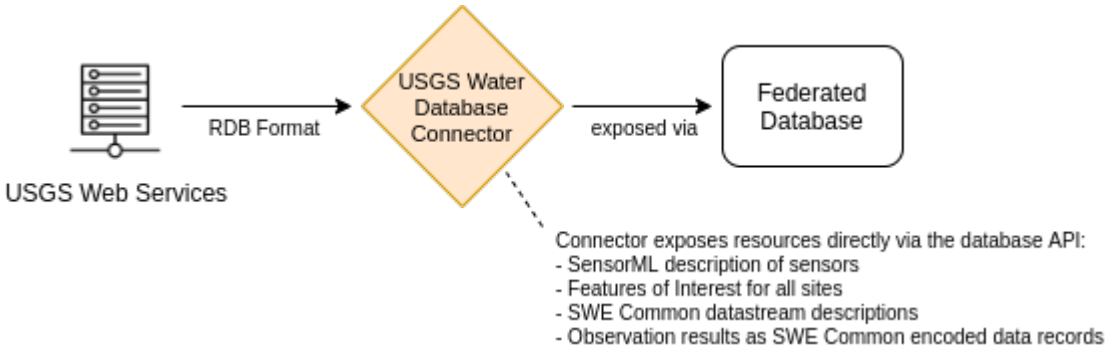

Figure 11 — USGS Water Database Connector Diagram

The following sensors & datastreams are made available by this connector:

SYSTEM TYPE	DATASTREAMS	COMMENTS
Water Monitoring Site	Discharge, Gage Height, Water Temperature, Wind Speed, Wind Direction, Ocean Surface Elevation, …	See the complete list of parameters. Note that sites typically don't report values for all parameters.

This connector demonstrates the feasibility of integrating large sensor networks and expose the corresponding metadata and observation data efficiently via OGC APIs. The connector was instantiated as a direct pass-through proxy (i.e. without caching) for the Testbed.

Individual stations provide in-situ measurements at fixed locations so they are modeled as features of interest using O&M `SF_SamplingPoint`. A different datastream is created for each site/parameter combination.

Future work on this connector is to better model the different site types (atmosphere, glacier, ocean/costal/estuary, lake, stream, spring, well, etc.) and expose metadata of each site using SensorML (for now, sites are only described using simple features of interest).

8.8. Processing

Processing chains are also a point of focus of this implementation. OSH natively supports real-time and batch processing and uses SensorML to provide a complete description of the processing chains.

The processing chains presented below take raw observations as input (data extracted from MISB video stream in this case) and produce new observations as their output. These new "derived" or "processed" observations are modeled and made available by OSH through the same APIs as the raw observations.

8.8.1. Image Frame Georeferencing

This process chain computes the ground coordinates of the area imaged by the sensor. The process outputs the geographic coordinates of each image corner as projected on the ground, taking into account the location and orientation of the image sensor, the platform attitude, and the sensor field of view.

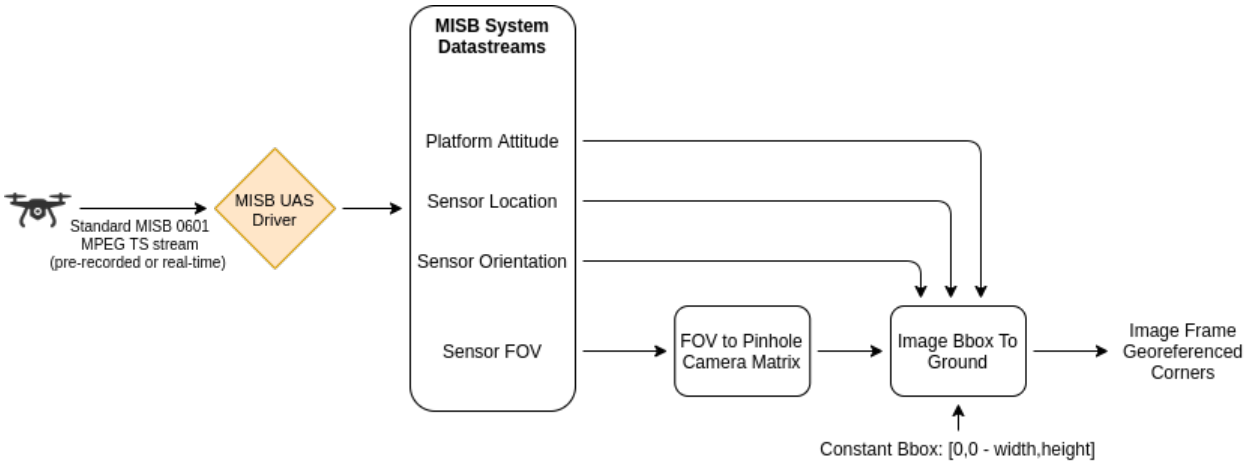

Figure 12 — Image Corners Georerefencing Process Chain

For reference, the full SensorML description of this processing chain is available from the demo server in both XML and JSON formats.

8.8.2. Video Object Tracking and Geolocation

This processing chain computes the ground location of one or more targets detected and tracked in the video. The process outputs the geographic coordinates of the targets for each video frame where they are detected. This processing chain takes the video frames as input and must also take into account all positioning parameters provided as part of the MISB stream.

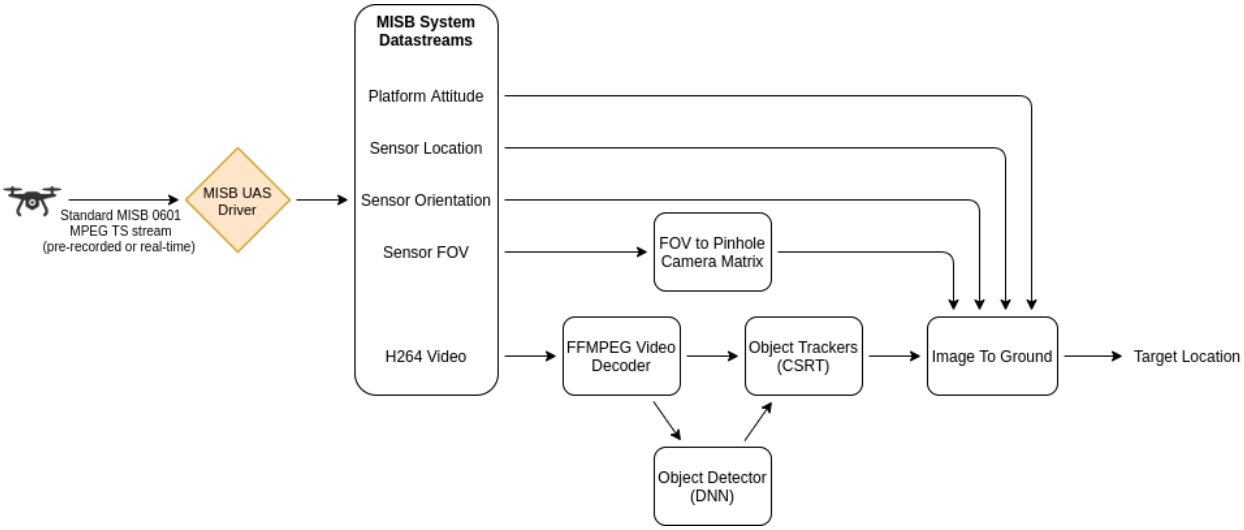

Figure 13 — Target Geolocation Process Chain

The processing chain description (in SensorML) is a good example of why concepts defined in the SIF semantic framework (such as different kinds of reference systems, orientation, and location data) and robustly provided by OGC SWE encoding standards are important to unambiguously define relationships between various system components.

For reference, the full SensorML description of this processing chain is available from the demo server in both XML and JSON formats.

8.9. Web Services and APIs

Several web services and APIs are part of the OSH based test implementation, exposing system metadata and observation data in various ways, and allowing the sensor hub to act as a bridge between various standard protocols (see Clause 8.4). This section provides some implementation details as well as the strengths and weaknesses of each of these interfaces.

8.9.1. Sensor Observation Service v2.0

OSH can expose observation data, system metadata (i.e. sensor system, models, etc.) and features of interest via the SOS 2.0 protocol. OSH also supports inserting system metadata and observations via the transactional SOS-T protocol.

8.9.1.1. Implementation Details

OSH implements SOS by exposing all or a subset of the systems, datastreams, and observations registered on a given sensor hub. A filter is used to select which resources are exposed via the SOS interface.

The following mappings are applied between OSH internal unified model (see Clause 8.3) and SOS requests:

SOS READ OPERATION	HOW IT IS MAPPED TO THE UNIFIED MODEL
GetCapabilities	1. List all System resources selected to be exposed via SOS. 2. For each System, create an ObservationOffering with the list of all ObservableProperties available through the System's Datastreams.
DescribeSensor	Retrieve SensorML System details.
GetFeature OfInterest	1. Find Datastreams matching the selected procedures, offerings and observed properties. 2. Retrieve features of interest that are the target of observations in these Datastreams.
GetObservation	1. Find Datastreams matching the selected procedures, offerings and observed properties. 2. Retrieve full observations from these Datastreams, applying the temporal and spatial filters.
GetResult Template	Find the Datastream matching the selected offering and observable property and return its SWE Common schema.
GetResult	Same as GetObservation except only observation results are returned as SWE Common encoded records.

SOS TRANSACTIONAL OPERATION	HOW IT IS MAPPED TO THE UNIFIED MODEL
InsertSensor	1. Create a new System. 2. Create a new FeaturesOfInterest if included in the observation template.
DeleteSensor	Delete the System referenced by ID, as well as all associated resources (subsystems, Datastreams, Observations, etc.)
UpdateSensor	Update the SensorML description of the System referenced by ID.
InsertResult Template	Create a Datastream attached to the previously created System.
Insert Observation	1. Find the Datastream matching the selected offering and observable property combination. 2. Create an Observation associated to this Datastream and the specified FoI. 3. Create a new FeatureOfInterest if provided inline with the Observation and it doesn't exist yet.
Delete Observation	Delete the observation referenced by ID.
InsertResult	Same as InsertObservation with the assumption that the FoI stays the same.

8.9.1.2. Strengths & Weaknesses

Strengths

- Standard support for binary encoded observations via GetResult (e.g. video frames).
- Fully aligned with O&M and SensorML since they are the default formats for observations and procedure descriptions respectively.

Weaknesses

- Older XML web service, harder to understand than REST.
- No support for streaming or pub/sub in the standard, although a websocket extension has been added by several implementations (OSH, MASBUS).
- SOS is not well suited for discovery if a large number of offerings/layers are provided in the capabilities document (i.e. there is no means to filter offerings/layers listed on a capabilities document).

8.9.2. Sensor Planning Service v2.0

OpenSensorHub can receive commands through the SPS 2.0 protocol. Commands are then forwarded to system drivers that in turn convert the commands to the system's own (legacy) protocol.

8.9.2.1. Implementation Details

OSH implements SPS by exposing commands supported by all or a subset of the systems registered on a given sensor hub. A filter is used to select which resources are exposed via the SPS interface.

The following mappings are applied between OSH internal unified model (see Clause 8.3) and SPS requests:

SPS REQUEST	HOW IT IS MAPPED TO THE UNIFIED MODEL
GetCapabilities	1. List all System resources selected to be exposed via SPS. 2. For each System, create a SensorOffering with a choice of all CommandStreams.
DescribeSensor	Retrieve SensorML System details.
DescribeTasking	Find the CommandStream matching the selected offering and return its SWE Common schema.
GetFeasibilty	Not implemented.
Submit	1. Find the CommandStream matching the selected offering/procedure. 2. Create a Command associated to this CommandStream and return its taskID.

SPS REQUEST	HOW IT IS MAPPED TO THE UNIFIED MODEL
GetStatus	Retrieve the current status of the Command corresponding to the provided taskID.
GetTask	Retrieve the complete Command corresponding to the provided taskID.
Update	Update the parameters of the Command corresponding to the provided taskID.

8.9.2.2. Strengths & Weaknesses

Strengths

- Standard support for binary encoded commands, which allows to send larger data blocks in-band as command inputs (e.g. an image).
- Fully aligned with SensorML since it is the default formats for procedure descriptions.

Weaknesses

- Older XML web service, harder to understand than REST.
- No support for streaming or pub/sub in the standard, although a websocket extension has been implemented as part of OSH.
- SPS is not well suited for discovery if a large number of offerings/layers are provided in the capabilities document (i.e. there is no means to filter offerings/layers listed on a capabilities document).

8.9.3. SensorThings API v1.0

OSH can expose observation data as well as sensor & actuator metadata via an implementation of the SensorThings API 1.0 standard (both HTTP/REST and MQTT are supported). An OSH based implementation of the SensorThings API Tasking Extension is under development.

8.9.3.1. Implementation Details

OSH implements SensorThings by exposing all or a subset of the systems, datastreams and command streams registered on a given sensor hub. A filter is used to select which resources are exposed via the SensorThings API.

The OSH SensorThing implementation is based on OSH internal unified model, with the following design decisions:

- STA Things are modeled as OSH Systems.

- STA Sensors and Actuators are modeled as OSH Systems and are members (i.e. subsystems) of the System representing the Thing.
- STA Locations and HistoricalLocations are modeled as observations in a dedicated Datastream attached to the System/Thing.

This is illustrated on the following diagram comparing instances of the SensorThings resource model and OSH internal model:

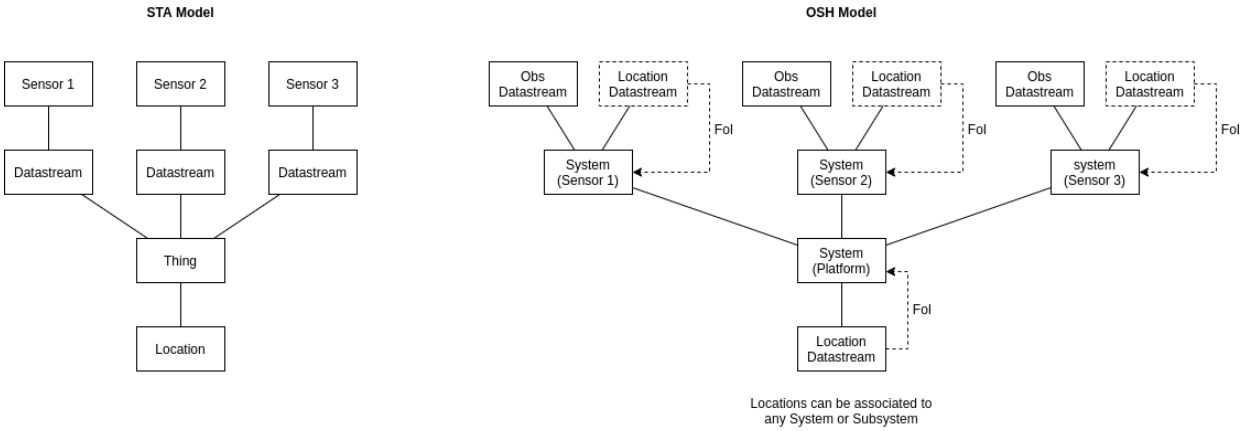

Figure 14 — Things vs. Systems

The following mappings are applied between OSH internal unified model (see Clause 8.3) and SensorThings requests:

OPERATIONS ON THINGS	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve the list of systems that have more than one member (i.e. systems with subsystems) or that are not tagged as either SSN Sensor or SSN Actuator + add a Thing representing the SensorHub that all “orphan” sensors will be attached to.
POST	Create a System with the provided name and description.
PUT/PATCH	Update the System description.
DELETE	Remove a System and all associated resources (Datastreams, Observations, CommandStreams, Commands).

OPERATIONS ON LOCATIONS	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve the latest observation from the location Datastream of the System representing the STA Thing.
POST	Add an Observation with the current time to the location Datastream of the System representing the STA Thing.

OPERATIONS ON LOCATIONS	HOW IT IS MAPPED TO THE UNIFIED MODEL
PUT/PATCH	Update the Observation whose ID is the same as the Location ID.
DELETE	Delete the Observation whose ID is the same as the Location ID.

OPERATIONS ON HISTORICALLO(	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve all observations from the location Datastream of the System representing the STA Thing.
POST	Add an Observation with the provided time to the location Datastream of the System representing the STA Thing.
PUT/PATCH	Update the Observation whose ID is the same as the HistoricalLocation ID.
DELETE	Delete the Observation whose ID is the same as the HistoricalLocation ID.

OPERATIONS ON SENSORS	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve all Systems that have no members and at least one Datastream or that are tagged as SSN Sensor.
POST	Create a System and semantically tag it with SSN Sensor class.
PUT/PATCH	Update the description of the System with the given ID.
DELETE	Remove the System with the given ID and all nested resources (i.e. subsystems, datastreams, command streams).

OPERATIONS ON DATASTREAMS	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve the list of all Datastreams that are part of Systems exposed by the service and whose result type is a scalar data type (i.e. boolean, category, count or quantity).
POST	Add a Datastream to the System represented by the STA Sensor that the new STA Datastream must refer to.
PUT/PATCH	Update the Datastream with the given ID. Reject request if the Datastream contained observations and the updated result structure is different from the existing one.
DELETE	Delete the Datastream with the given ID.

OPERATIONS ON MULTIDATASTR	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve the list of all Datastreams that are part of Systems exposed by the service and whose result type is a record (exclude all Datastreams with more complex results such as arrays).
POST	Add a Datastream to the System represented by the STA Sensor referenced by the STA Datastream.
PUT/PATCH	Update the Datastream with the given ID. Reject if the Datastream contained observations and the updated result structure is different from the existing one.
DELETE	Delete the Datastream with the given ID.

OPERATIONS ON OBSERVATIONS	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve Observations from all Datastreams exposed by the service.
POST	Add an Observation to the Datastream referenced by ID by the STA Observation.
PUT/PATCH	Update the Observation with the given ID.
DELETE	Delete the Observation with the given ID.

OPERATIONS ON OBSERVEDPRO	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Not implemented
POST	Not implemented
PUT/PATCH	Not implemented
DELETE	Not implemented

OPERATIONS ON FEATUREOFINT	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve the list of all FeaturesOfInterest
POST	Add a new FeatureOfInterest.

OPERATIONS ON FEATUREOFINT	HOW IT IS MAPPED TO THE UNIFIED MODEL
PUT/PATCH	Update the FeatureOfInterest with the given ID.
DELETE	Delete the FeatureOfInterest with the given ID.

OPERATIONS ON ACTUATORS	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve all Systems that have no members and at least one CommandStream or that are tagged as SSN Actuator.
POST	Create a System and semantically tag it with SSN Actuator class.
PUT/PATCH	Update the System description.
DELETE	Remove the System and all associated resources.

OPERATIONS ON TASKINGCAPAB	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve the list of all CommandStreams that are part of Systems exposed by the service.
POST	Add a CommandStream to the System represented by the STA Actuator referenced by the STA Datastream.
PUT/PATCH	Update the CommandStream with the given ID. Reject if the CommandStream has already received commands and the updated parameter structure is different from the existing one.
DELETE	Delete the CommandStream with the given ID.

OPERATIONS ON TASKS	HOW IT IS MAPPED TO THE UNIFIED MODEL
GET	Retrieve Commands from all CommandStreams exposed by the service.
POST	Add a Command to the CommandStream referenced by ID by the STA Task.
PUT/PATCH	Update the Command with the given ID.
DELETE	Delete the Command with the given ID.

8.9.3.2. Strengths & Weaknesses

Strengths

- A simple API that is easy to understand from the client point of view (although not necessarily easy to implement on the server side).
- Suited for discovery even if large collections are used.
- Covers both data access and tasking through a single API.

Weaknesses

- Difficult to model datastreams with vector observed properties (e.g. location, orientation vector or quaternions, etc.).
- Lack of support for binary data prevents the use of this interface to handle high bandwidth sensors such as video or point clouds. In this case, linking to out-of-band data or service endpoints is usually necessary.
- Modeling more complex systems and relations between their components is difficult. For example, Actuator and Sensor are separate entities, making it hard for sensors to accept commands and vice-versa. Datastreams are attached to both a Sensor and a Thing but there is no concept of system hierarchy where the sensor can be part of a larger system or platform.
- Missing a standard way of doing simple keyword or full-text search.

8.9.4. SensorWeb API (Draft)

OpenSensorHub also includes an API under development called the SensorWeb API that could be of interest for the OGC community. This API provides more or less direct access to OSH unified model, with the following features:

- Discovery and access to the full hierarchy of Systems and their Subsystems, with full historization (i.e. historical system descriptions and configurations are also accessible so that they can be used to interpret historical observations correctly).
- Multi format and encoding support. Each Datastream describes its record structure as well as encoding allowing streaming observations as compressed imagery, video or audio for example (see Clause 8.13 for details).
- Full text search in addition to parameter search.
- Query DSL for JOIN queries.

- In addition to REST, support for Websocket and MQTT for streaming observations and other resource events (i.e. a system is added, deleted, changed, etc.).
- Local property ontology enabling the definition of properties specific to a particular system but still relying on higher level ontologies (like W3C SSN, QUDT, SensorML ontology).
- The API is designed to be federated (i.e. a hub can aggregate data from other hubs and expose it like its own).

More details about the SensorWeb API are provided in Annex B.

8.10. Modeling of Features of Interest

Features of Interest (FoI) are an essential part of the O&M model because they provide more information about the object that is being observed. Features of Interest are often physical entities (i.e. real-world objects, geographic features, etc.) but can also be more abstract entities. They can also be an intermediary object (proxy feature of interest) that samples or represents a larger entity (ultimate feature of interest).

Below are several examples of possible features of interest:

- A natural geographic feature such as a river
- A man-made geographic feature such as a building
- A monitoring station on a river (i.e. 2D sampling point of a larger river feature)
- A well sampling an underground aquifer (i.e. 3D sampling point of a larger aquifer feature)
- The volume of space surrounding a radiation sensor (i.e. 3D sampling volume)
- A mobile ground vehicle (i.e. mobile sampling point)
- An aircraft or UAV
- The exact area imaged by a street camera (i.e. sampling of a larger road feature)
- The area of the earth surface sampled by a remote sensor (e.g. polygonal sampling surface)
- A financial asset (e.g. for which we observe price and transaction volume)

Particular attention should be given to the modeling of features of interest in the case of certain mobile and remote sensors. Difficulties typically arise in these cases because the sampling geometry varies quickly with time, sometimes at the same rate as the observations themselves (think of video cameras providing 30 observations per second or more). Note that geometry

can vary, not only because of a change of location, but also because of a change of sensor orientation or other parameters (e.g. the field of view of an imager). In such cases, providing the sampling geometry as part of the feature of interest object at each time instant is not only highly inefficient but also make discovery much more difficult.

Another approach is to separate constant (or rarely updated) feature properties from the ones that are highly variable. Noting that highly variable feature properties are often the ones that are observed continuously and constant ones are often asserted or observed only once (because they are known to be fixed a-priori), it seems only natural to model variable properties as time series of observations. Following such an approach, the OSH model can capture constant properties inside the feature resource itself, while variable property values are captured as separate Datastreams of Observations. In more complex cases, the sampling geometry is not directly measured but rather computed from simpler observations

Note that there are still some use cases whereby creating a different feature of interest for each observation is desirable. This is typically true when each feature is considered to have its own identity. For example, images of the earth taken by a space-borne still imager can be associated with a different feature every time with its own identifier and sampling geometry. However, this typically does not make sense for video cameras that tend to image the same objects multiple times per second. Providing a different feature along with every single video frame would be confusing.

The table below provides several examples of relationships between Sensor Systems, their Datastreams and Features of Interest demonstrated during the testbed:

SENSOR SYSTEM	OBS DATASTREAM	FEATURE OF INTEREST OR SAMPLING FEATURE	SAMPLED FEATURE	SAMPLING GEOMETRY DATASTREAM
Fixed Radiation Sensor	Radiation Reading	Sphere centered at the sensor location and radius equal to the detection range	The feature or general area where the sensor is located (e.g. building, street or geographic area)	-
Fixed Radiation Sensor	Radio Link Status	Sensor itself	-	-
Mobile Radiation Sensor	Radiation Reading	Sphere centered at the sensor location and radius equal to the detection range (shape property links to sampling geometry datastream)	The feature or general area where the sensor is being used (e.g. geographic area where the mobile sensor is usually deployed)	Sensor Location
Mobile Radiation Sensor	Sensor Location	Sensor itself	-	-
Mobile Radiation Sensor	Radio Link Status	Sensor itself	-	-

SENSOR SYSTEM	OBS DATASTREAM	FEATURE OF INTEREST OR SAMPLING FEATURE	SAMPLED FEATURE	SAMPLING GEOMETRY DATASTREAM
MISB UAS	UAS Video	Dynamic SamplingFeature representing the Imaging Area. Instantaneous sampling geometry provided separately.	General area where mission takes place	Geo-Referenced Image Frame
MISB UAS	Platform Attitude	Platform itself	-	-
MISB UAS	Platform Location	Platform itself	-	-
MISB UAS	Camera Orientation	Sensor itself	-	-
MISB UAS	Camera Parameters (FOVs)	Sensor itself	-	-

8.11. Relative Positioning

SWE Common Vector and Matrix constructs are useful to describe any kind of vector and tensor quantities. In particular, they allow the provision of necessary metadata when dealing with relative positioning of remote sensing system components (including both location and orientation).

A SWE Common vector allows specifying both a reference frame (the frame or CRS with respect to which the position is provided) and a local frame (the frame whose position is provided by the vector). This allows providing the relative position between subsystems unambiguously. The following snippets show how reference frames are specified in a SWE Common Vector.

Example 1:

Location of sensor with respect to a geodetic coordinate reference system (here EPSG 9705):

```
{
    "type": "Vector",
    "label": "Sensor Location",
    "definition": "http://www.opengis.net/def/property/OGC/0/SensorLocation",
    "referenceFrame": "http://www.opengis.net/def/crs/EPSG/0/9705",
    "localFrame": "urn:osh:sensor:uas:predator001#SENSOR_FRAME",
    "coordinates": [
        latitude ...,
        longitude ...,
        altitude ...
    ]
}
```

Example 2:

Attitude (orientation) of platform with respect to a local North-East-Down frame:

```
{
```

```
    "type": "Vector",
    "label": "Platform Attitude",
    "definition": "http://www.opengis.net/def/property/OGC/0/
PlatformOrientation",
    "referenceFrame": "http://www.opengis.net/def/cs/OGC/0/NED",
    "localFrame": "urn:osh:sensor:uas:predator001#PLATFORM_FRAME",
    "coordinates": [
        heading ...,
        pitch ...,
        roll ...
    ]
}
```

Example 3:

Orientation of sensor with respect to the platform it is mounted on:

```
{
    "type": "Vector",
    "label": "Sensor Orientation",
    "definition": "http://www.opengis.net/def/property/OGC/0/
SensorOrientation",
    "referenceFrame": "urn:osh:sensor:uas:predator001#PLATFORM_FRAME",
    "localFrame": "urn:osh:sensor:uas:predator001#SENSOR_FRAME",
    "coordinates": [
        heading ...,
        pitch ...,
        roll ...
    ]
}
```

See Annex A.2.2 for full examples.

8.12. Semantics

Metadata provided by OSH typically makes use of SWE Common which separates structure from semantics, allowing referencing semantical concepts in external dictionaries, taxonomies, or ontologies.

For example, the following snippet shows the description of a data field (for example a field that is part of an observation result) that is a quantity expressed in `hPa` and represents the value of “air pressure at cloud top” as defined by the CF dictionary:

```
{
    "name": "ctp",
    "type": "Quantity",
    "label": "Cloud Top Pressure",
    "description": "Atmospheric pressure at cloud top",
    "definition": "http://mmisw.org/ont/cf/parameter/air_pressure_at_cloud_
top",
    "uom": {
        "code": "hPa"
    }
}
```

8.13. Data Encoding

SWE Common defines the data encoding separately from the structure and semantics. This allows the same data to be encoded in different ways while keeping the structure/schema unchanged. The following encoding methods are supported:

- DSV / CSV (DSV = delimiter separated value; CSV is a particular case where the separator is a comma)
- XML
- JSON
- Binary

The SWE Common schema is provided as part of the Datastream description (exposed via SOS GetResultTemplate or via the SensorWeb API Datastream schema resource). Observation results can then be encoded efficiently using just the values.

8.13.1. Binary Encoding and Codecs

The binary encoding supports wrapping well-known codecs via the BinaryBlock construct. This allows describing the structure as if the data were provided in "decoded" form, but still encode the data in efficient ways.

Example: An image is an 2D array of pixels where each pixel has N-channels. The image will be described as so using SWE Common DataArray construct even though it is provided in a compressed form as a binary block in the actual datastream.

In principle, any codecs are supported by specifying a proper mimetype or URI for the codec in the compression attribute of the BinaryBlock. For example, the following compression, video and audio codecs have been implemented in OpenSensorHub:

Generic compression:

COMPRESSION TYPE	IDENTIFICATION STRING
GZIP	application/gzip
BZIP	application/x-bzip2

For video and audio codecs, mime types or web media-type codec parameter strings are used. The prefixes for these strings are provided below:

Video Codecs:

CODEC TYPE	IDENTIFICATION STRING
H264	video/h264 or avc1,...
H265 / HEVC	video/hevc or hevc,...
VP8	vp08,...
VP9	vp09,...
VP10	vp10,...
JPEG / MJPEG	image/jpeg

Audio Codecs:

CODEC TYPE	IDENTIFICATION STRING
WAV / PCM	audio/pcm
AAC	aac,...
OPUS	opus,...
VORBIS	vorbis,...

Note that these are codecs and not media container formats such as MP3, MP4, AVI, WebM, MKV, etc. Instead, the SWE Common datastream acts as the container. However, in addition to making video and audio data available via standard SWE interfaces, OpenSensorHub is also capable of acting as a video server by wrapping the coded frames into a container format such as MP4 on the fly.

A

ANNEX A (INFORMATIVE) JSON ENCODING OF SWE DATA MODELS

A ANNEX A (INFORMATIVE) JSON ENCODING OF SWE DATA MODELS

A.1. JSON Schemas

Draft JSON schemas for SWE Common Data Model v2.0 are available at the following location:

https://github.com/opensensorhub/sensorweb-api/blob/master/swe-common_v20_schema.json

Draft JSON schemas for SensorML v2.1 are currently under development and available at the following location:

https://github.com/opensensorhub/sensorweb-api/blob/master/sensorml_v20_schema.json

A.2. JSON Examples

A.2.1. ISA SensorML Example

```
{
  "type": "PhysicalSystem",
  "id": "mrlxembvu1kj",
  "definition": "http://www.w3.org/ns/sosa/Sensor",
  "identifier": "urn:osh:sensor:isa:701149:RADIO003",
  "names": ["Radiological Sensor RADIO003"],
  "identifications": [
    {
      "type": "IdentifierList",
      "identifiers": [
        {
          "type": "Term",
          "definition": "http://sensorml.com/ont/swe/property/Manufacturer",
          "label": "Manufacturer Name",
          "value": "Radiological Sensors, Inc."
        },
        {
          "type": "Term",
          "definition": "http://sensorml.com/ont/swe/property/ModelNumber",
          "label": "Model Number",
```

```
        "value": "RD123"
      },
      {
        "type": "Term",
        "definition": "http://sensorml.com/ont/swe/property/SoftwareVersion",
        "label": "Software Version",
        "value": "FW21.23.89"
      }
    ]
  }
],
"classifications": [
  {
    "type": "ClassifierList",
    "classifiers": [
      {
        "type": "Term",
        "definition": "http://sensorml.com/ont/swe/property/SensorType",
        "label": "Sensor Type",
        "value": "RadiologicalSensor"
      }
    ]
  }
],
"validTimes": [
  {
    "type": "TimePeriod",
    "id": "T1",
    "beginPosition": "2021-05-20T11:56:43.444Z",
    "endPosition": {
      "indeterminatePosition": "now"
    }
  }
],
"characteristics": [
  {
    "name": "operating",
    "type": "CharacteristicList",
    "definition": "http://www.w3.org/ns/ssn/systems/OperatingRange",
    "label": "Operating Characteristics",
    "characteristics": [
      {
        "name": "voltage",
        "type": "Quantity",
        "definition": "http://qudt.org/vocab/quantitykind/Voltage",
        "label": "Operating Voltage",
        "uom": {
          "code": "V"
        },
        "value": 12.0
      },
      {
        "name": "current",
        "type": "Quantity",
        "definition": "http://qudt.org/vocab/quantitykind/Current",
        "label": "Operating Current",
        "uom": {
          "code": "A"
        },
        "value": 1.5
      },
      {
        "name": "batt_capacity",
```

```
          "type": "Quantity",
          "definition": "http://sensorml.com/ont/swe/property/BatteryCapacity",
          "label": "Battery Capacity",
          "uom": {
            "code": "W.h"
          },
          "value": 200.0
        }
      ]
    }
  ],
  "capabilities": [
    {
      "name": "measurement",
      "type": "CapabilityList",
      "definition": "http://www.w3.org/ns/ssn/systems/SystemCapability",
      "label": "System Capabilities",
      "capabilities": [
        {
          "name": "resolution",
          "type": "Quantity",
          "definition": "http://www.w3.org/ns/ssn/systems/Resolution",
          "label": "Resolution",
          "uom": {
            "code": "[ppm]"
          },
          "value": 1.0
        },
        {
          "name": "accuracy",
          "type": "Quantity",
          "definition": "http://sensorml.com/ont/swe/property/
AbsoluteAccuracy",
          "label": "Absolute Accuracy",
          "uom": {
            "code": "[ppm]"
          },
          "value": 10.0
        },
        {
          "name": "integ_time",
          "type": "Quantity",
          "definition": "http://sensorml.com/ont/swe/property/IntegrationTime",
          "label": "Integration Time",
          "uom": {
            "code": "s"
          },
          "value": 14.0
        },
        {
          "name": "sampling_freq",
          "type": "Quantity",
          "definition": "http://sensorml.com/ont/swe/property/
SamplingFrequency",
          "label": "Sampling Frequency",
          "uom": {
            "code": "Hz"
          },
          "value": 0.0
        }
      ]
    }
  ]
```

```
  ...
}
```

A.2.2. MISB Datastreams Examples

```
{
  "datastream": "gal7w6j6v7n9",
  "name": "Predator UAS (MISB) - Sensor Location",
  "resultSchema": {
    "type": "DataRecord",
    "label": "Sensor Location",
    "description": "Geographic location of imaging sensor accounting for lever
 arm between the platform gps antenna and the sensor",
    "fields": [
      {
        "name": "time",
        "type": "Time",
        "definition": "http://www.opengis.net/def/property/OGC/0/SamplingTime",
        "referenceFrame": "http://www.opengis.net/def/trs/BIPM/0/UTC",
        "label": "Precision Time Stamp",
        "uom": {
          "href": "http://www.opengis.net/def/uom/ISO-8601/0/Gregorian"
        }
      },
      {
        "name": "location",
        "type": "Vector",
        "definition": "http://www.opengis.net/def/property/OGC/0/
SensorLocation",
        "referenceFrame": "http://www.opengis.net/def/crs/EPSG/0/9705",
        "localFrame": "#SENSOR_FRAME",
        "coordinates": [
          {
            "name": "lat",
            "type": "Quantity",
            "definition": "http://sensorml.com/ont/swe/property/
GeodeticLatitude",
            "axisID": "Lat",
            "label": "Geodetic Latitude",
            "uom": {
              "code": "deg"
            }
          },
          {
            "name": "lon",
            "type": "Quantity",
            "definition": "http://sensorml.com/ont/swe/property/Longitude",
            "axisID": "Lon",
            "label": "Longitude",
            "uom": {
              "code": "deg"
            }
          },
          {
            "name": "alt",
            "type": "Quantity",
            "definition": "http://sensorml.com/ont/swe/property/
HeightAboveMSL",
            "axisID": "h",
            "label": "MSL Height",
            "uom": {
```

```
              "code": "m"
            }
          }
        ]
      }
    ]
  },
  "resultEncoding": {
    "type": "TextEncoding",
    "collapseWhiteSpaces": "true",
    "decimalSeparator": ".",
    "tokenSeparator": ",",
    "blockSeparator": "\n"
  }
}
```

B

ANNEX B (INFORMATIVE) SENSORWEB API (DRAFT)

ANNEX B (INFORMATIVE) SENSORWEB API (DRAFT)

B.1. OpenAPI Specification

The OpenAPI document for the SensorWeb API (draft) is available at:

https://github.com/opensensorhub/sensorweb-api/blob/master/sensorweb-api.yaml

There is also a test page available at:

https://opensensorhub.github.io/sensorweb-api/swagger-ui

C

ANNEX C (INFORMATIVE) REVISION HISTORY

C

ANNEX C (INFORMATIVE) REVISION HISTORY

DATE	RELEASE	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
June 10, 2021	0.1	A. Robin	all	Initial ER version
Sep 09, 2021	0.2	A. Robin	all	Draft ER version
Dec 16, 2021	1.0	A. Robin	all	Final version for public release