OGC® DOCUMENT: 21-029

External identifier of this OGC® document: http://www.opengis.net/doc/PER/t17-D031

Open Geospatial Consortium

OGC TESTBED 17: MASBUS INTEGRATION ENGINEERING REPORT

ENGINEERING REPORT

PUBLISHED

Submission Date: 2021-11-19
Approval Date: 2021-12-09
Publication Date: 2022-03-31
Editor: Dr. Sara Saeedi

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

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) analyses the Measures and Signatures Intelligence Enterprise Service Bus (MASBUS) pilot software and the efforts to integrate with OGC SensorThings API resources. After introducing MASBUS, a server implementation is designed to digest sensor data and demonstrate the SensorThings MQTT (Message Queuing Telemetry Transport) extension of the MASBUS software. To show the SensorThings MQTT extension of the MASBUS software, a MASBUS client implementation is also presented. This ER discusses the results of the MASBUS integration, including all lessons learned from the experiments completed during the OGC Testbed 17 Sensor Integration thread and concludes with a set of optimum recommendations.

II EXECUTIVE SUMMARY

Sensor systems are built using many different standards, formats, and protocols. This variety results in a significant barrier to sensor integration. Sensor standards must embrace this diversity. What is required is not a one-size-fits-all solution, but a framework of standards that enable integration of sensors regardless of the technical constraints.

This Testbed-17 MASBUS Integration Engineering Report (ER) describes how interoperability can be established in a common scenario of heterogeneous data sources (e.g. sensors, IoT platforms, simulators and other data models and encodings). Moreover, the architecture presented in this ER allows querying and visualizing observations from data sources using widely adopted international standards such as the OGC SensorThings API (OGC 19-088).

The ER concludes that MASBUS integrated implementation involves complex distributed data models and systems with multiple diverse applications. To make well-informed decisions, it is essential to have a proper data integration strategy, which must allow working with heterogeneous data sources and platforms in interoperable ways. The ER presents a series of findings regarding the server-side and client-side components.

The ER recommends the following activities for further work in future OGC Testbeds and Standards Working Groups:

- Development of the draft Sensor Web API through alignment with both OGC API Common and the OGC SensorThings API. Such alignment could also involve support for the Observations, Measurements, and Samples (OMS) 3.0 candidate standard.
- Advancement of the JSON Encoding Rules SWE Common / SensorML Best Practice (OGC 17-011r2) towards adoption as an OGC Standard.
- Explore the potential use of the Sensor Model Register provided by the OGC Naming Authority (OGC-NA) to deliver semantic definitions in support of MASINT use cases.

This could potentially lead to a pan update to the OGC-NA policy on Sensor Models and Parameters (OGC 18-042r4).

III KEYWORDS

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

ogcdoc, OGC document, MASBUS, SOS, SWE, sensor, API, MASINT

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):

- University of Calgary

VII SUBMITTERS

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

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

1

SCOPE

1 SCOPE

This OGC Testbed 17 Engineering Report (ER) analyses the Measures and Signatures Intelligence Enterprise Service Bus (MASBUS) pilot software and the efforts to integrate with OGC SensorThings API resources. After introducing MASBUS, a server implementation is designed to digest sensor data and demonstrate the SensorThings MQTT (Message Queuing Telemetry Transport) extension of the MASBUS software. To show the SensorThings MQTT extension of the MASBUS software, a MASBUS client implementation is also presented. This ER discusses the results of the MASBUS integration, including all lessons learned from the experiments completed during the OGC Testbed 17 Sensor Integration thread and concludes with a set of optimum recommendations.

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

2.1.1. **Measurement**

A set of operations to determine the value of a quantity. (OGC 10-004r3 / ISO 19156:2011)

2.1.2. **MISB-compliant UAS**

Unmanned Air System that is compliant with Motion Imagery Standards Board standards

2.1.3. **Observation**

Act of measuring or otherwise determining the value of a property. (OGC 10-004r3 / ISO 19156:2011)

2.1.4. 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)

2.1.5. Sensor Integration Framework (SIF)

a framework that provides the guidance required for sensor data producers and consumers to implement a sensor information enterprise that meets operational requirements (Source: https://github.com/ngageoint/Sensor_Integration_Framework)

2.2. Abbreviated terms

- **API** Application Programming Interface
- **CSV** Comma Separated Value
- **CWS** Catalogue Service for the Web
- **D&I** Defense and Intelligence
- **DC** Direct Current
- **DDF** Data Framework References)
- **DDMS** Defense Discovery Metadata Standard
- **DMO** Defense MASINT Office
- **DOD/IC** Department of Defense/Intelligence Community
- **DI2E** Defense Intelligence Information Enterprise
- **DSILKRI** DI2E Sensor Information Loading Kit Reference Implementation
- **DSOSRI** DI2E Sensor Observation Service Reference Implementation
- **DSV** Delimiter Separated Value
- **ER** Engineering Report
- **ETL** Extract-Transform-Load

- **FoI** Feature of Interest
- **ID** Identity
- **ISA** Integrated Sensor Architecture
- **JSON** JavaScript Object Notation
- **Lat/Lon** Latitude/Longitude
- **LRU** Least Recently Used
- **MASBUS** Measures and Signatures Intelligence Enterprise Service Bus
- **MASINT** Measurement and Signature Intelligence
- **MIME** Multipurpose Internet Mail Extensions
- **MISB** Motion Imagery Standards Board
- **MQTT** Message Queuing Telemetry Transport
- **MSL** Mean Sea Level Height
- **NED** North-East-Down
- **NATO** North Atlantic Treaty Organization
- **OGC** Open Geospatial Consortium
- **OMS** Observations, Measurements and Samples
- **OSH** Open Sensor Hub
- **Pub/Sub** Publish/Subscribe
- **RDF** Resource Description Framework
- **REST** Representational State Transfer
- **SIF** Sensor Integration Framework
- **SOS** Sensor Observation Service
- **SPS** Sensor Planning Service
- **SSN** Semantic Sensor Network
- **STA** SensorThings API
- **SWE** Sensor Web Enablement
- **TB-17** Testbed-17
- **TIE** Technology Integration Experiments

- **UAS** Unmanned Aircraft System
- **URI** Uniform Resource Identifier
- **USGS** US Geological Survey
- **XML** Extensible Markup Language

3

OVERVIEW

3 OVERVIEW

The Engineering Report (ER)discusses MASBUS, MASBUS SensorThings Integration, client-server architecture, and the MASBUS client for accessing stored sensor data. Then, the results of the MASBUS SensorThings Integration experiments for online performance are evaluated. Finally, the recommendations based on the work completed in this testbed task are presented.

The following figure (Figure 1) describes the MASBUS integration workflow executed in Testbed-17.

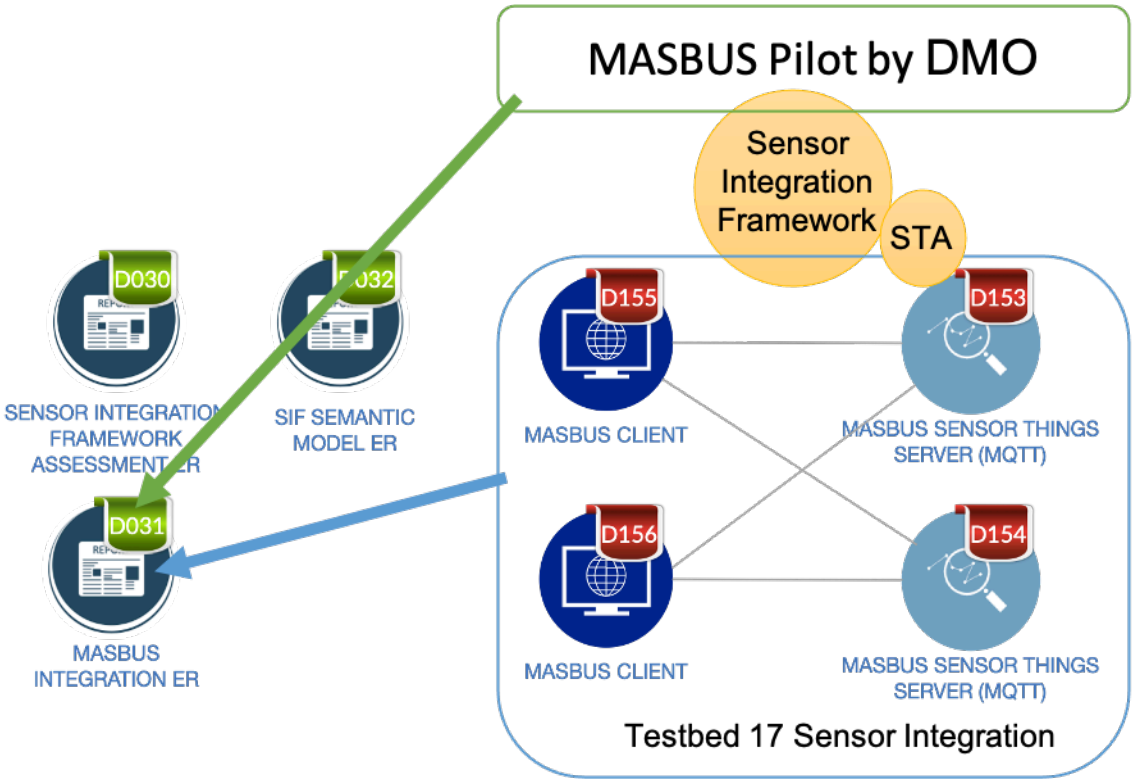

Figure 1 — MASBUS integration ER and workflow

This document contains the following sections:

- **Chapter 4 — MASBUS Background:** This section includes the Background and review of the MASBUS Pilot.
- **Chapter 5 — MASBUS Server Instance:** This section describes MASBUS SensorThings Integration. In this section, SensorThings API (STA) integration to the MASBUS is discussed including System Architecture and Components. Two main components of this architecture, Server and Client Components, are discussed in detail.
- **Chapter 6 – MASBUS Client Instance:** This section describes the testing of MASBUS STA Web Applications. After presenting the main components, the MASBUS STA Web Application is tested in a scenario.

- **Chapter 7 — Technology Integration Experiments:** This section documents the Technology Integration Experiments (TIEs) conducted as part of this Testbed-17 MASBUS task.
- **Chapter 8 — Findings:** This section presents the findings of the research.

Unresolved directive in document.adoc — include::/builds/ogc/T17-D031-MASBUS-Integration-ER/sources/sections/05-MASBUS-Background.adoc[]

4

MASBUS SERVER INSTANCE

4 MASBUS SERVER INSTANCE

This section, which is also presented in the Testbed-17 Sensor Integration Framework Assessment ER (Deliverable D030), discusses the server implementation based on OpenSensorHub (OSH) that was developed during Testbed-17 to demonstrate the feasibility of SIF concepts.

The MASBUS SensorThings data model is shown in Figure 2.

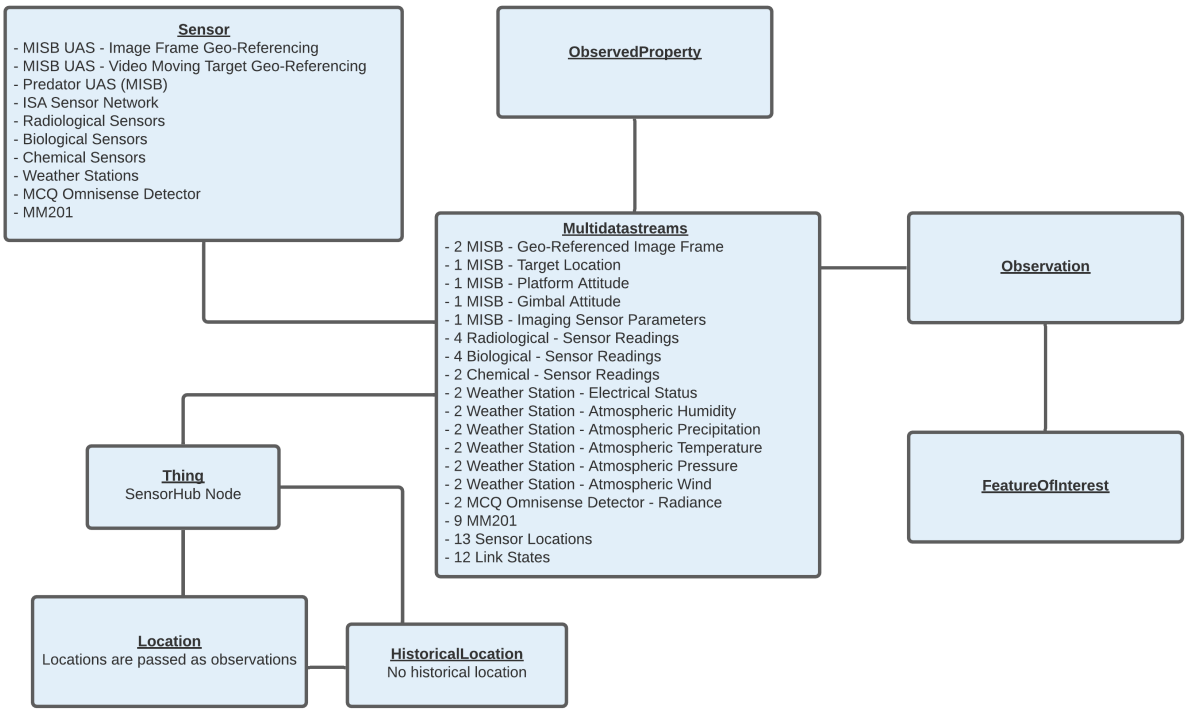

Figure 2 — MASBUS SensorThings MQTT extension - SensorThings Data Model

4.1. 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., a 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 in Figure 3.

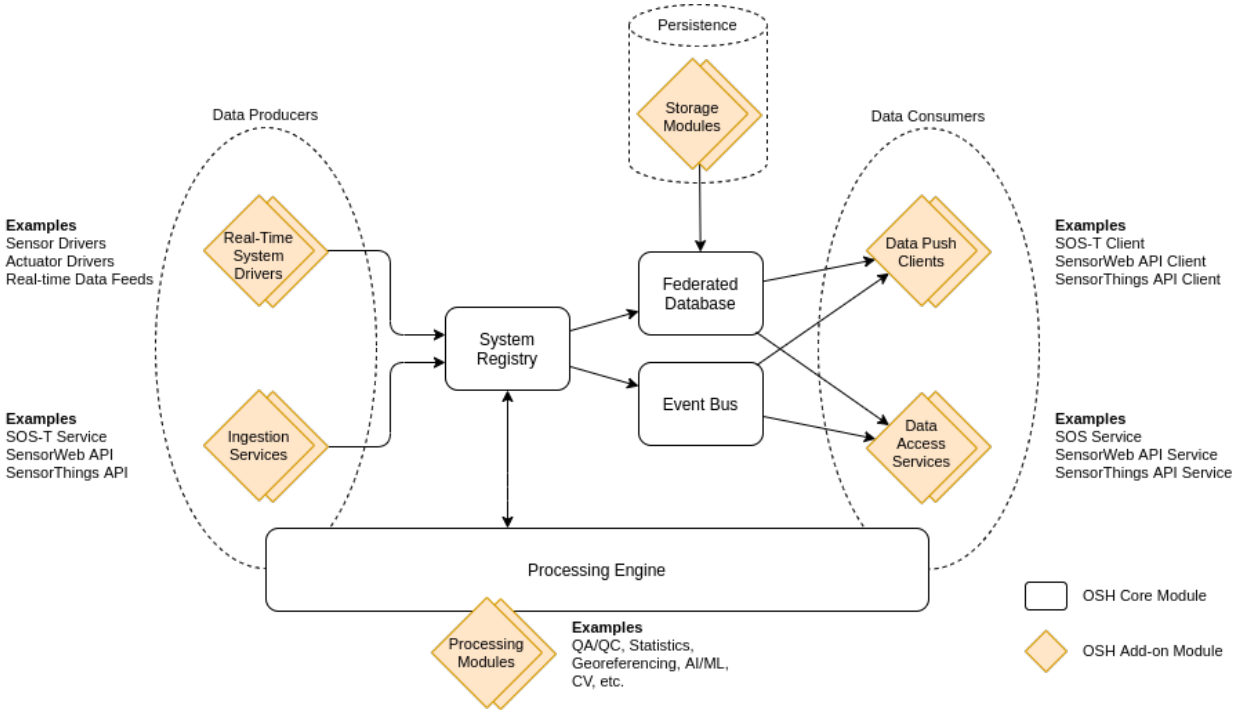

Figure 3 — OSH Architecture

OpenSensorHub is built on conceptual models defined in Observations & Measurements (O&M) (OGC 10-004r3), Sensor Model Language (SensorML) (OGC 12-000r2) and SWE Common standards (OGC 08-094r1). OSH also implements the JSON Encoding Rules of SWE Common / SensorML (OGC 17-011r2), as well as the next version of the O&M standard, which will be called Observations, Measurements, and Samples (OMS). Additional semantics are provided by Semantic Sensor Network (SSN) ontology as well as the SensorML ontology browsable at http://sensorml.com/ont.

4.2. Multi-protocol Bridge

OSH implements the concept of a protocol bridge envisioned by SIF. Ingesting data using one protocol and transforming it with another is possible. OSH performs the proper transformation and tries to minimize the loss of information in the process. For example, the following transformations were demonstrated in the Testbed 17 initiative:

- Ingest data from a standard SOS server and expose the data through SensorThings API and SensorWeb API endpoints.
- Ingest data using SOS-T and expose the data through SensorThings API and SensorWeb API endpoints.
- Ingest data using SensorThings API endpoint and expose the data via SOS and SensorWeb API endpoints.

- Ingest Motion Imagery Standards Board (MISB) full motion video data/metadata and expose the data/metadata via SOS, SensorThings API and SensorWeb API endpoints.
- Ingest Integrated Sensor Architecture (ISA) protocol buffer data and expose via SOS, SensorThings API and SensorWeb API endpoints.
- Ingest data from legacy US Geological Survey (USGS) web services and expose the data via SOS, SensorThings API and SensorWeb API endpoints.

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

4.3. Integration Points

OpenSensorHub (OSH) can integrate with a legacy system in different ways:

1. 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 drivers support 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 converting 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.
2. Another way of integrating an external device or other real-time data feed is by developing a software component that runs separately from OSH and exchanges data with it 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, the extension can be developed in any programming language. This method can also be used to ingest large amounts of historical data in batch mode.
3. Finally, integrating directly with an existing datastore containing historical observations and/or metadata by using the Datastore API is possible. In this case, OSH acts as a proxy to an underlying datastore with or without maintaining its own cache (e.g., Least Recently Used (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 the Testbed 17 initiative.

4.4. 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., the Integrated Sensor Architecture (ISA)). In Testbed 17, the following drivers were developed and showcased: ISA Protocol Driver and MISB UAS Driver.

4.4.1. ISA Protocol Driver

The Integrated Sensor Architecture (ISA) enables sensors of various kinds to be dynamically discovered and commanded, as well as to efficiently exchange information over networks. The ISA Protocol Driver is used to connect systems that implement the ISA protocol, and has two modes of operation:

1. Connection to real-time ISA sensor feed supporting the Protocol Buffer (Protobuf) encoding.
2. 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.

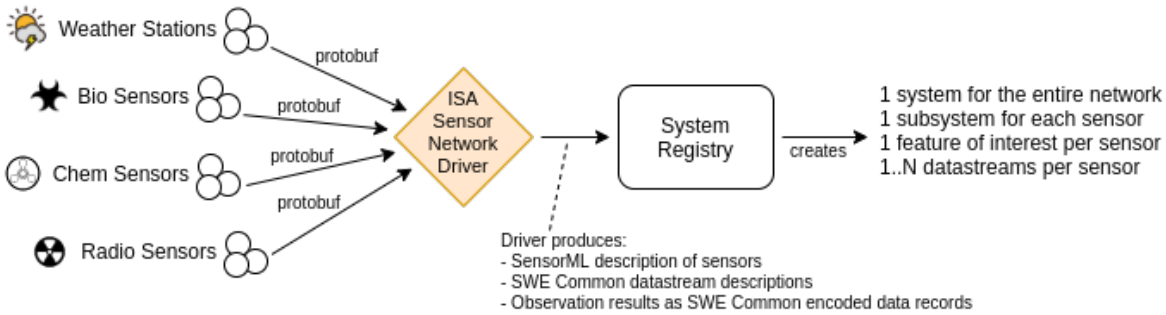

Figure 4 — ISA Driver Diagram

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 (JavaScript Object Notation) encoding is provided in Annex A.2.1 of the Testbed 17 Sensor Integration Framework Assessment ER (OGC 21-022). Note that properties are semantically tagged using concept URIs that are resolvable to external ontologies. It could be possible in the future to reference concepts that are registered in the Sensor Model Register of the OGC Naming Authority (OGC-NA). The OGC-NA Sensor Model Register is currently at prototype stage.

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), Sensor Location	BIO001 to BIO004 in the demonstration, with simulated random measurements
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

4.4.2. 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).

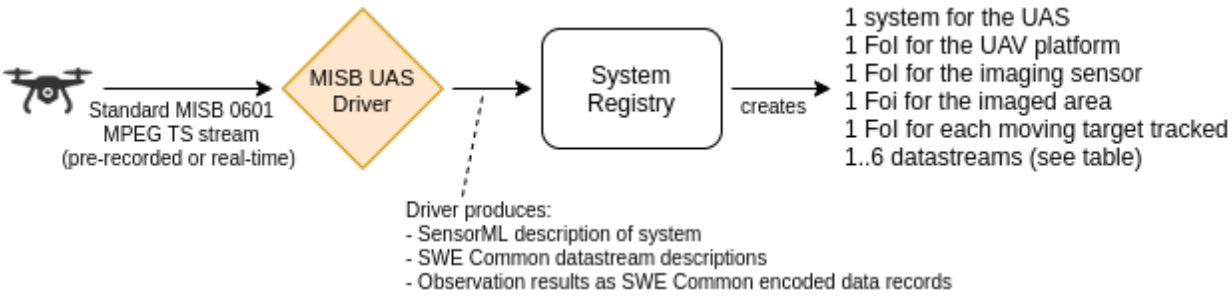

Figure 5 — MISB Driver Diagram

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 FOVs
Georeferenced Image Frame	Imaged Area	Geographic coordinates (EPSG 4979)
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:

1. Be able to identify and describe the feature of interest for each datastream.
2. 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 4.9 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 sea level (MSL).
- The local horizontal North-East-Down (NED) reference frame.
- The platform reference frame rigidly attached to the aircraft/UAV.
- The sensor reference frame 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 of the Testbed-17 Sensor Integration Framework Assessment ER (OGC 21-022). Note that properties are semantically tagged using concept uniform resource identifiers (URIs) that are resolvable to external ontologies.

4.5. Datastore Connectors

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

- MASBUS SOS server, using the standard OGC SOS protocol;
- USGS Water Database, using the legacy web service protocol and RDB (tab delimited) format

4.5.1. 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.

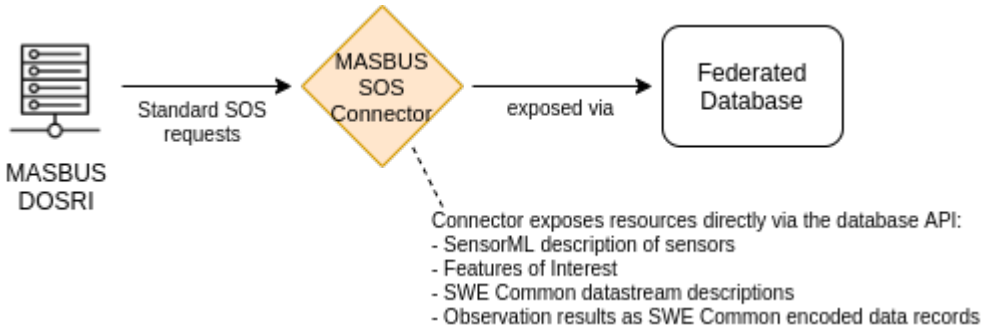

Figure 6 — MASBUS Connector Diagram

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*`

4.5.2. 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.

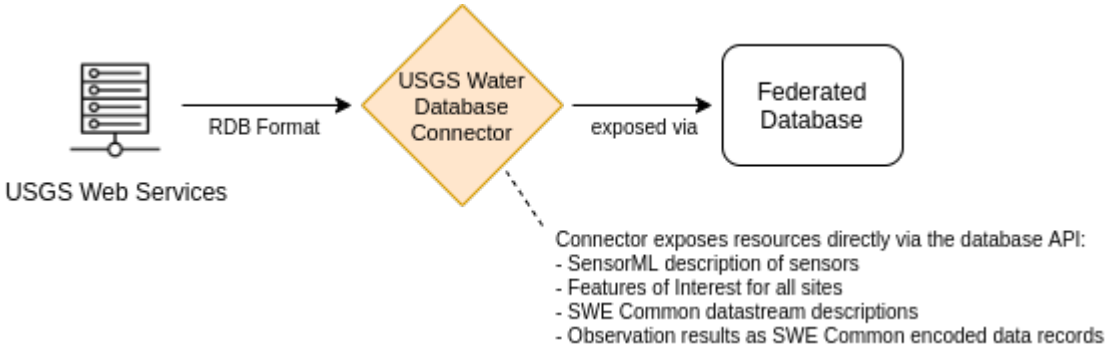

Figure 7 — USGS Water Database Connector Diagram

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 exposes the corresponding metadata and observation data efficiently via implementations of OGC Standards. 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 modelled 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/coastal/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).

4.6. 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.

4.6.1. Image Frame Georeferencing

This process chain computes the ground coordinates of the area imaged by the sensor. Image georeferencing 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.

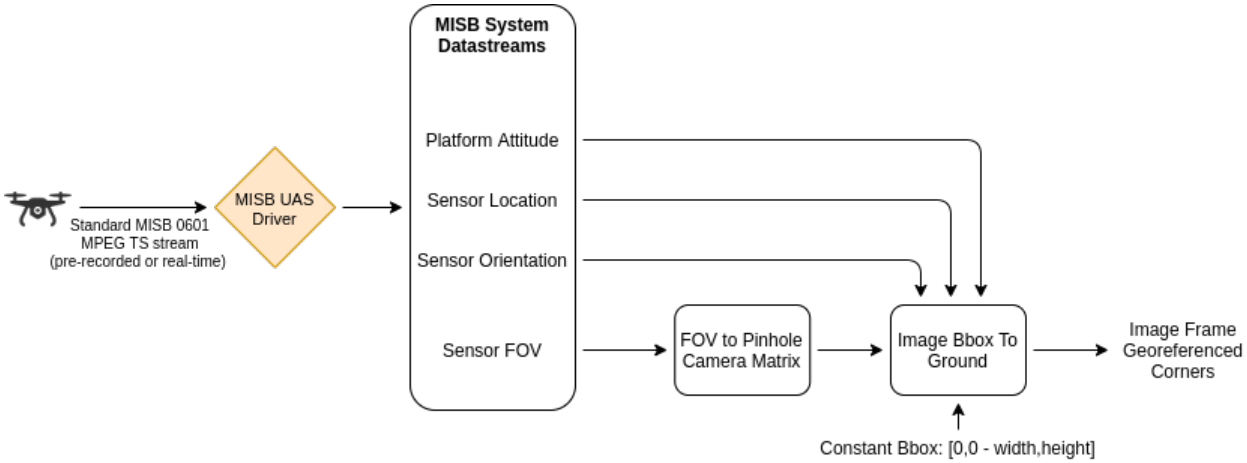

Figure 8 — Image Corners Georerefencing Process Chain

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

4.6.2. 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.

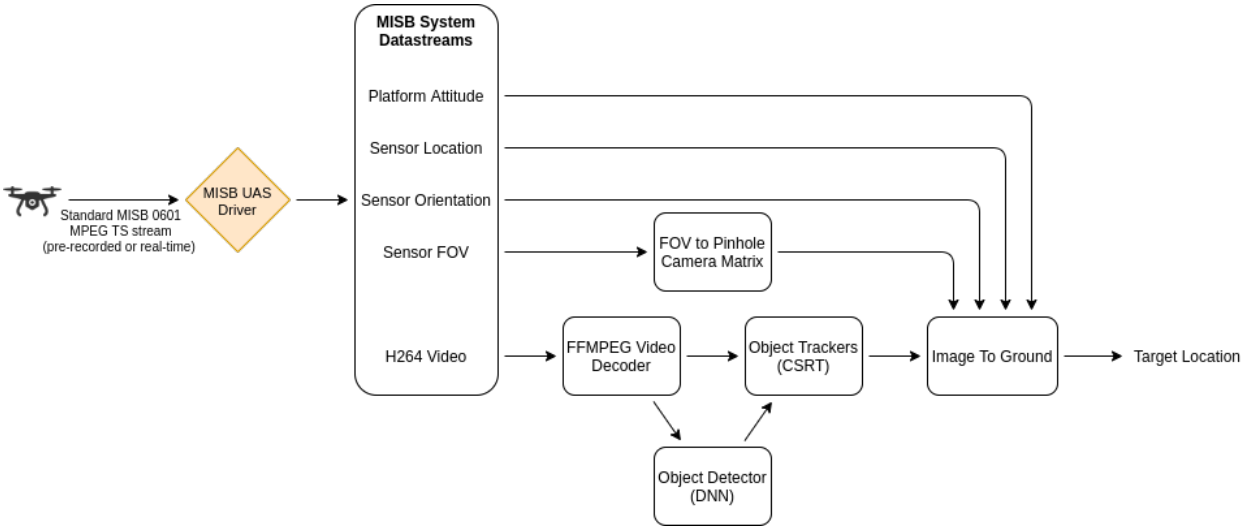

Figure 9 — Target Geolocation Process Chain

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.

4.7. Web Services and APIs

Several web services and APIs are part of the OSH-based test implementation, supporting the ability to expose system metadata and observation data in various ways, and enabling the sensor hub to act as a bridge between various standard protocols (see Clause 4.2). //??ref This section provides some implementation details as well as the strengths and weaknesses of each of these interfaces.

4.7.1. Sensor Observation Service v2.0

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

4.7.1.1. 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 the OSH internal unified model (see Section 8.3 of OGC 21-022) 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 TRANSACTIONA 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.

SOS TRANSACTIONAL OPERATION	HOW IT IS MAPPED TO THE UNIFIED MODEL
InsertResult	Same as InsertObservation with the assumption that the FoI stays the same.

4.7.1.2. Strengths & Weaknesses

Strengths

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

Weaknesses

- Older XML (Extensible Markup Language) web service, harder to understand than a JSON-based RESTful service.
- No support for streaming or publish/subscribe (pub/sub) capabilities in the standard, although a WebSocket extension was added for several implementations (OSH, MASBUS).
- Like some OGC web services, SOS is not well suited for discovery if a large number of offerings are provided (i.e., no standard search capability).

4.7.2. Sensor Planning Service v2.0

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

4.7.2.1. 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 the OSH internal unified model (see Section 8.3 of OGC 21-022) 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.

SPS REQUEST	HOW IT IS MAPPED TO THE UNIFIED MODEL
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 with this CommandStream and return its taskID.
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.

4.7.2.2. Strengths & Weaknesses

Strengths

- Standard support for binary encoded commands, which supports sending larger data blocks in-band as command inputs (e.g., an image).
- Fully aligned with SensorML as the default format for procedure descriptions.

Weaknesses

- Older XML web service, harder to understand than a JSON-based RESTful service.
- No support for streaming or pub/subs in the standard, although a WebSocket extension was implemented as part of OSH.
- Like some other OGC web services, SPS is not well suited for discovery if a large number of offerings are provided (i.e., no standard search capability).

4.7.3. SensorThings v1.0

OSH can expose observation data as well as sensor & actuator metadata via the SensorThings 1.0 REST (Representational State Transfer) API. The tasking extension is under development.

4.7.3.1. Implementation Details

OSH implements the SensorThings API 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 SensorThings API implementation is based on the 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 the OSH internal model:

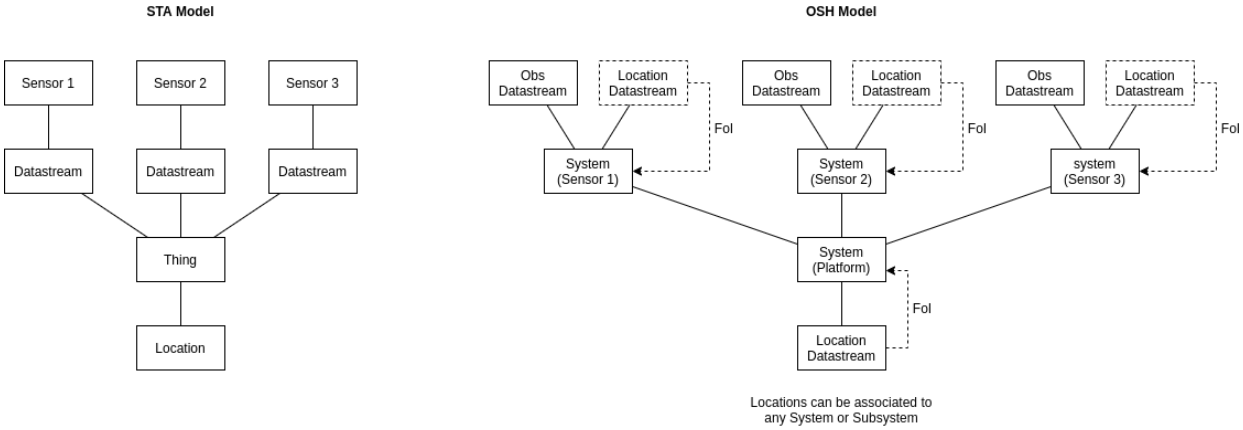

Figure 10 — Things vs. Systems

The following mappings are applied between the OSH internal unified model (see Section 8.3 of OGC 21-022) 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.

OPERATIONS ON THINGS	HOW IT IS MAPPED TO THE UNIFIED MODEL
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.
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 HISTORICALLOC	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 contains 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

OPERATIONS ON OBSERVEDPRO	HOW IT IS MAPPED TO THE UNIFIED MODEL
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.
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.

4.7.3.2. Strengths & Weaknesses

Strengths

- Simple API that's 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.
- Difficult to model more complex systems and relations between their components (e.g., Actuator and Sensor are separate entities which makes 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.

4.7.4. SensorWeb API

OSH also includes an API under development called the SensorWeb API that could be of interest to the OGC community. The API provides more or less direct access to the 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 4.11 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 at https://opensensorhub.github.io/sensorweb-api/swagger-ui/.

4.8. 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;

- 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 image). 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 makes 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 identifier and sampling geometry. However, this typically doesn't 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 feature or general area where	Sensor Location

SENSOR SYSTEM	OBS DATASTREAM	FEATURE OF INTEREST OR SAMPLING FEATURE	SAMPLED FEATURE	SAMPLING GEOMETRY DATASTREAM
		the detection range (shape property links to sampling geometry datastream)	the sensor is being used (e.g. , geographic area where the mobile sensor is usually deployed)	
Mobile Radiation Sensor	Sensor Location	Sensor itself	-	-
Mobile Radiation Sensor	Radio Link Status	Sensor itself	-	-
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	-	-

4.9. 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 supports 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 provides 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 EPSF 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/PlatformOrientatio
n",
    "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 of OGC 21-022 for full examples.

4.10. Semantics

Metadata provided by OSH typically makes use of SWE Common which separates structure from semantics so that referencing semantic concepts in external dictionaries, taxonomies or ontologies can be done.

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"
    }
}
```

4.11. Data Encoding

SWE Common defines the data encoding separately from the structure and semantics. As such, the same data can 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, with just the values.

4.11.1. Binary Encoding and Codecs

The binary encoding supports wrapping well-known codecs via the BinaryBlock construct. This enables describing the structure as if the data were provided in “decoded” form, but still encodes it in efficient ways.

Example: An image is a 2D array of pixels where each pixel has N-channels and is described 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 media type (Multipurpose Internet Mail Extensions: MIME type) or URI for the codec in the compression attribute of the BinaryBlock. For example, the following compression, video, and audio codecs are 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 `codecs` parameter strings is 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. The SWE Common datastream acts as the container. However, in addition to making video and audio data available via standard SWE interfaces, OSH is also capable of acting as a video server by wrapping the coded frames into a container format such as MP4 on the fly.

5

MASBUS CLIENT INSTANCE

5 MASBUS CLIENT INSTANCE

This section discusses the client implementation based of Testbed-17 MASBUS integration.

5.1. MASBUS 2D Client (University of Calgary)

The aim of the client is to test the server instance. The client architecture (see Figure 11) consists of the following main modules:

- *List of Topics*: a list of available MQTT topics for each Multidatastream is extracted.
- *MQTT Subscriber*: This module subscribes to all the topics available on the server, and receives MQTT payloads.
- *TL (Transform-Load)*: This module transforms the received MQTT payload into a consumable format for the client. This also includes handling the high sampling frequency of some Sensor readings. The sampling frequency of each Sensor reading is different. It can be as low as one sample every couple of days, for a biological sensor, or as high as 30 samples per second, for MISB drones. For the purpose of visualization, the high sampling frequencies greater than 1 Hz are decreased to 1 Hz.
- *Storing The Data*: This module is responsible for storing the transformed data for demonstrating historical datasets. It published the data to a corresponding MQTT Topic, which will be stored in an AWS S3 bucket.
- *Data Enrichment*: This module enriches each sensor reading with the location of the sensor, as locations are passed through different Multidatastreams on the server-side.

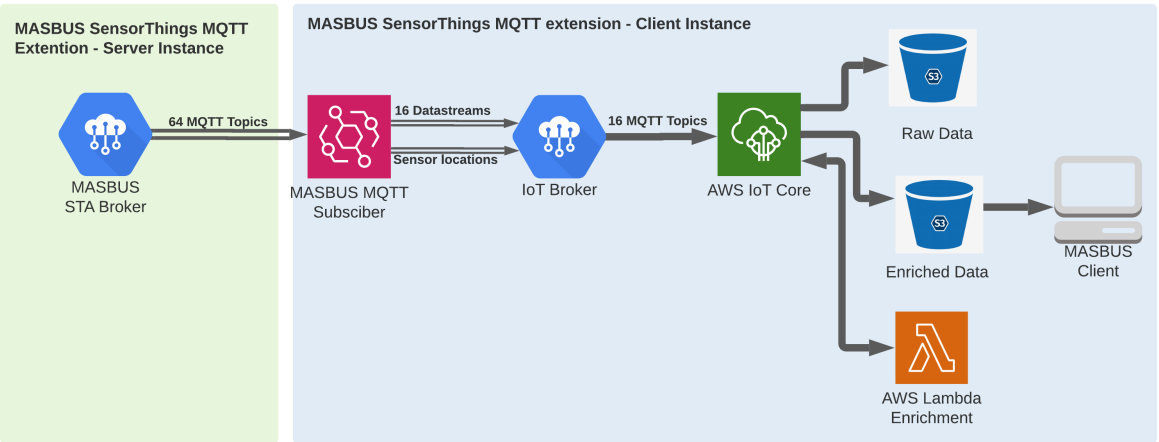

Figure 11 — MASBUS SensorThings MQTT extension - client architecture

5.1.1. MASBUS Client Data Enrichment

In the data model of the server, the location of each sensor is passed through a different Multidatastream. On the client-side, sensor readings are enriched with the most recent corresponding sensor location (Figure 12) and stored in an AWS S3 bucket.

Apart from the MISB Multidatastreams, all other sensors used in this testbed are stationary with a consistent location. But, only the location of MISB related Multidatastreams is published through MQTT. Therefore, on the client-side, the location of stationary Sensors is extracted via HTTP requests.

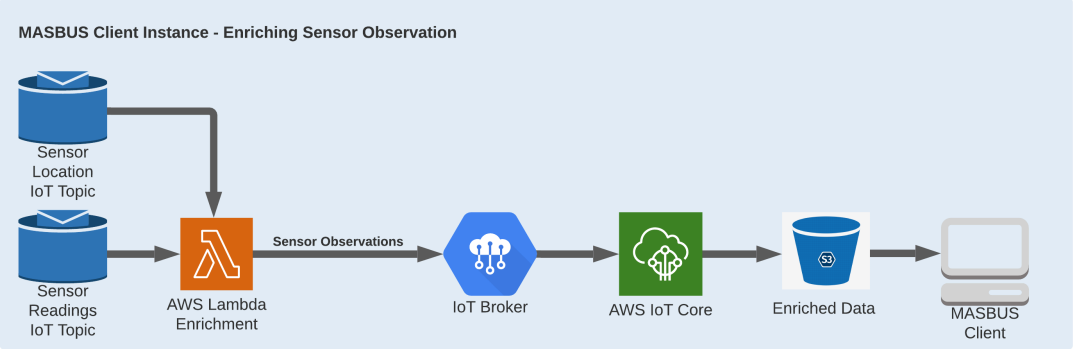

Figure 12 — MASBUS SensorThings MQTT extension - client architecture

5.1.2. MASBUS Client Features

This section is going to point out some of the Client's features.

5.1.2.1. Support Sensor Location and Observations In The Same Place

While the MASBUS server data model enquires different Multidatastreams for MISB sensor locations, on the client-side it is possible to enrich sensor readings with locations. Therefore, no additional layer will be required for demonstrating sensors on the map.

Also, a different dataset is dedicated to stationary sensor locations only, so that the user can see all the stationary sensors at the same time (Figure 13) on the map.

Figure 13 — All sensor locations in a single view

5.1.2.2. Filtering

- Multiple filters can be applied to the attributes of the dataset. The list of available filters depends on the type of attribute. Table 1 shows a list of available filters for the kind of attributes that were available in this testbed.

Table 1 — List of filters that can be applied on an attribute

ATTRIBUTE TYPE	FILTERS
String	- Equals — Does not equal — Contains — Does not contain — Is set
Number	- Equals — Does not equal — Is between — Is Greater than — Is less than — Is Greater than or equals — Is less than or equals

Figure 14 shows an example of filtering on sensor location.

Figure 14 — Filter sensor location based on the datastream name

This kind of filtering is applied to the retrieved data. There is another way for filtering a sensor name that would affect the query itself. See Figure 15.

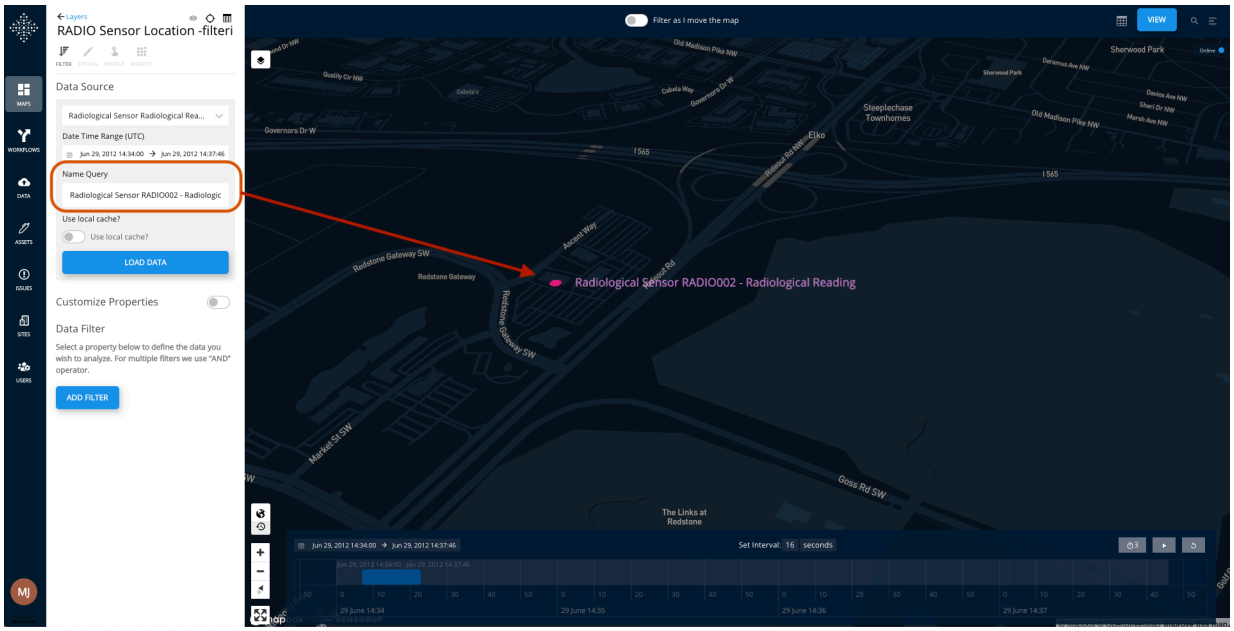

Figure 15 — Filter sensor location based on the datastream name

5.1.2.3. Table view and download the dataset

The client supports the table view of the data, as well as customizing the list of properties shown on the table, and downloading it as an xlsx file (Figure 16).

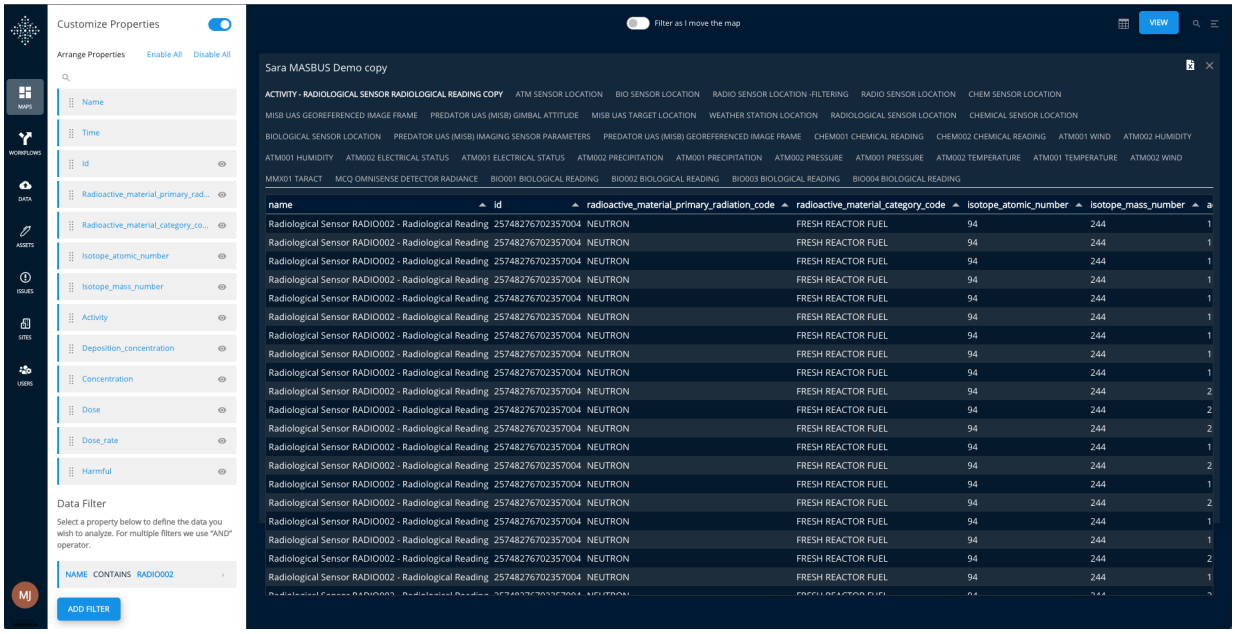

Figure 16 — Table view of the dataset

5.1.2.4. Supporting widget for each attribute

The client supports variant types of widgets listed in Table 2. An example of widgets for a temperature sensor is shown in Figure 17.

Table 2 — List of supported widgets

WIDGET TYPE	DESCRIPTION
Line Chart	A time-series line graph (X-Axis will be time in default)
Bar Chart	A bar graph where each feature will represent a bar where the data points are the aggregated observations.
Scatter Chart	In the scatter chart each two selected properties are plotted on a single graph, for example, the temperature values over time.
Numeric widget	For live data, it can show the last observation of each feature. In historical datasets, it shows a selected aggregation of all observations for each feature.
Gauge Chart	Represents the last observation of live data or a selected aggregation of all observations for each feature.

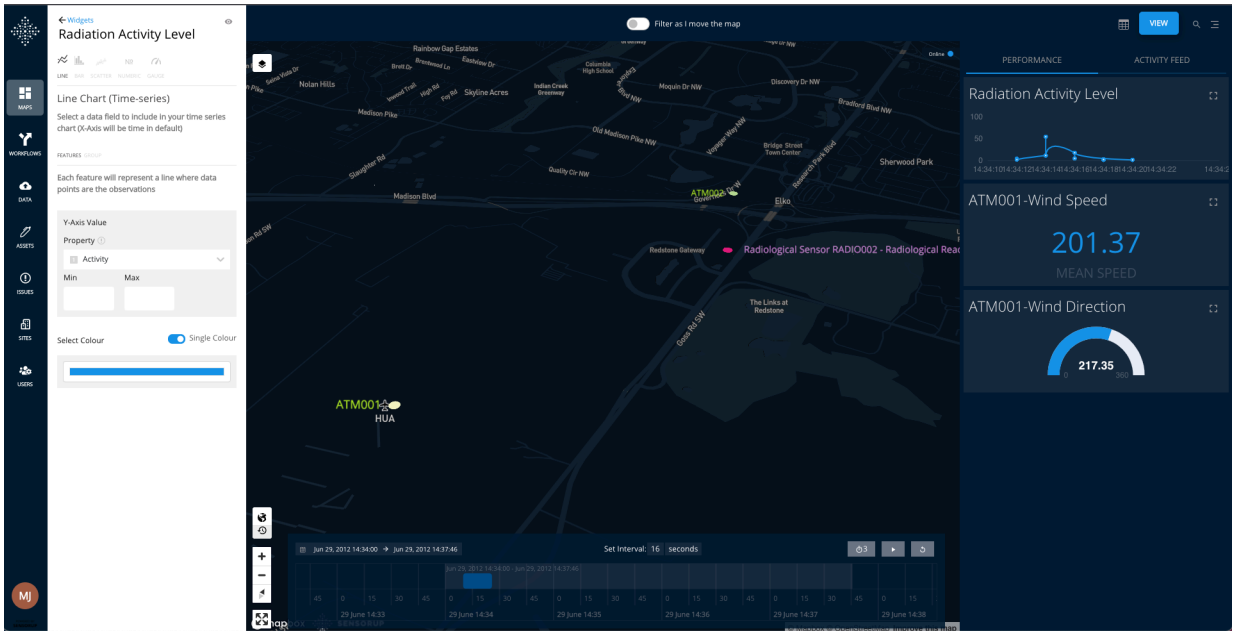

Figure 17 — Supported widgets on client

5.1.2.5. Customizable Pop-up Info Box

The list of attributes and their order are customizable on the client (Figure 18).

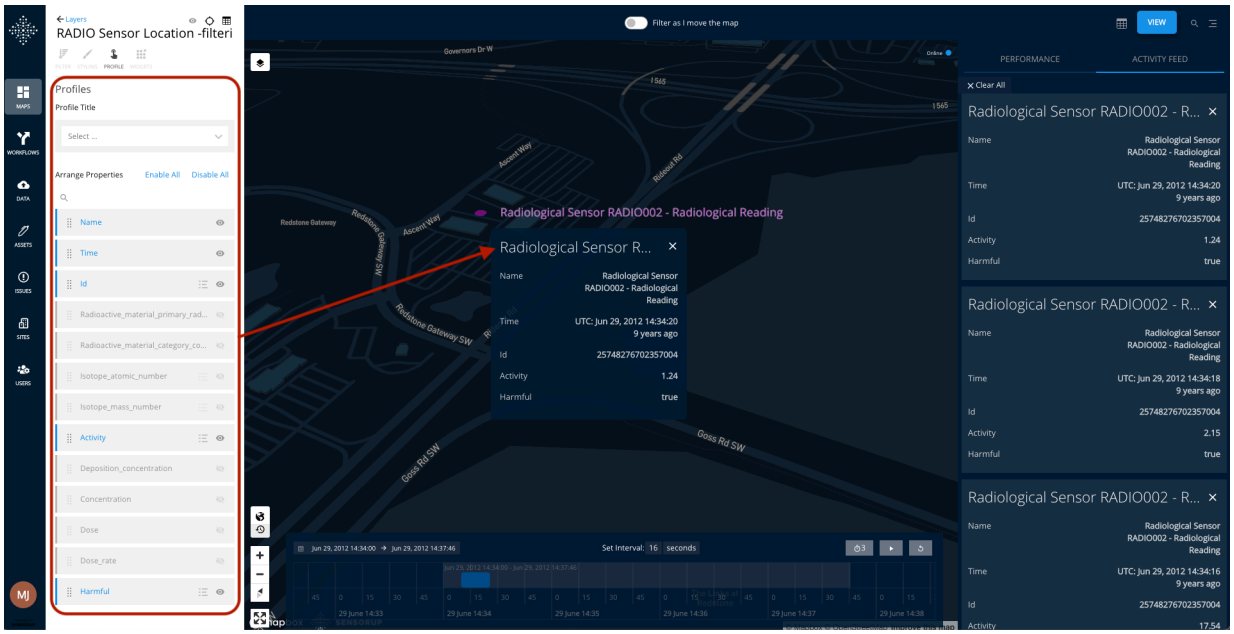

Figure 18 — Customizable popup info

5.1.2.6. Styling

The client supports different styling for demonstrating the data on the map. Some of the styling options are listed as follow:

- Representing the sensor as a point, or a custom icon with configurable color, opacity, radius, and visibility. The coloring can be based on an attribute value, and different colors can be assigned to different levels of values for an attribute. See Figure 19.
- Add a label based on an attribute with a configurable color, font size, text anchor, and alignment.
- Arc styling with adjustable height for LineString geometry type.
- Heatmap with a configurable set of colors, opacity, visibility, radius, intensity, weight based on an attribute, and level of smoothness at the boundaries of the heatmap.
- Hexbin with a configurable set of colors, opacity, visibility, hexagon radius, hexagon height based on an attribute.
- GeoJSON styling with adjustable filling color (simple or based on an attribute), line color (simple or based on an attribute), opacity, visibility, radius, height based on a numeric attribute, and line width.
- Polygon styling with a configurable set of color, opacity, visibility, and polygon height based on an attribute.

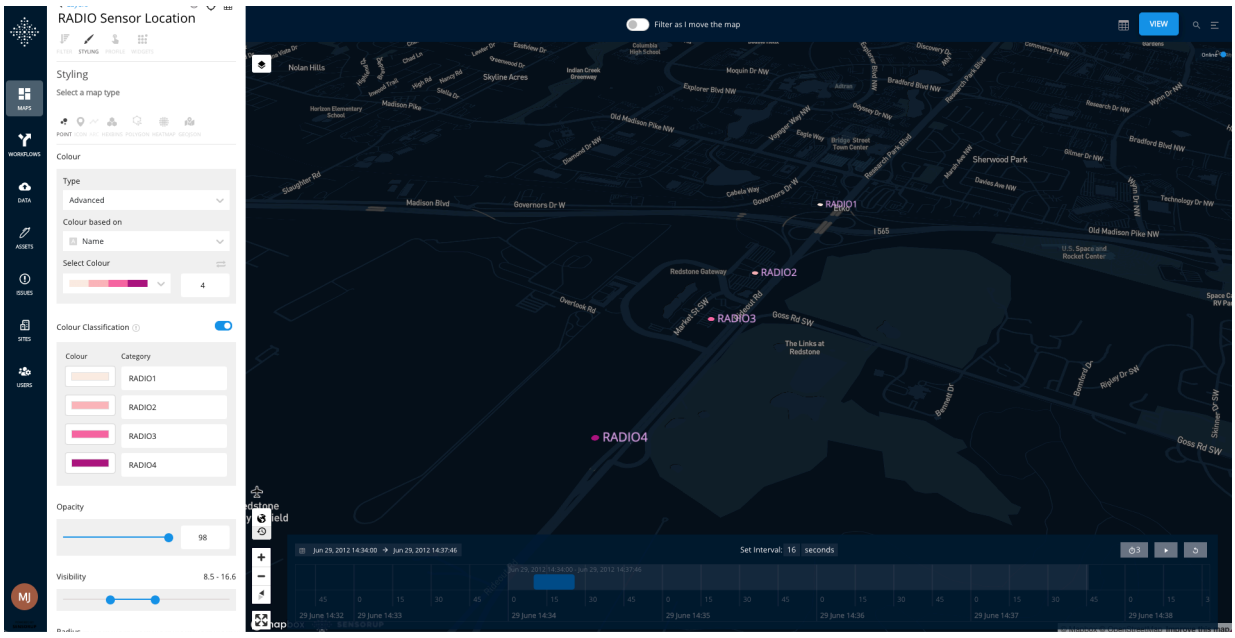

Figure 19 — Client Styling - Advanced Coloring

5.1.2.7. Client Workflow

The alerts can be generated using the `Workflow` feature of the client. The `Workflow` enables the user to apply a specific action (send an email, a text message, or create an issue with a due date and assign it to a particular person) if a condition is met. Figure 20 shows an example of generating alerts (emails, and text messages) on the `activity` attribute of the `Radiological Sensor - RADIO004`. By enabling this workflow, an alert will be generated whenever the `activity` level of `RADIO004` becomes greater than a threshold value.

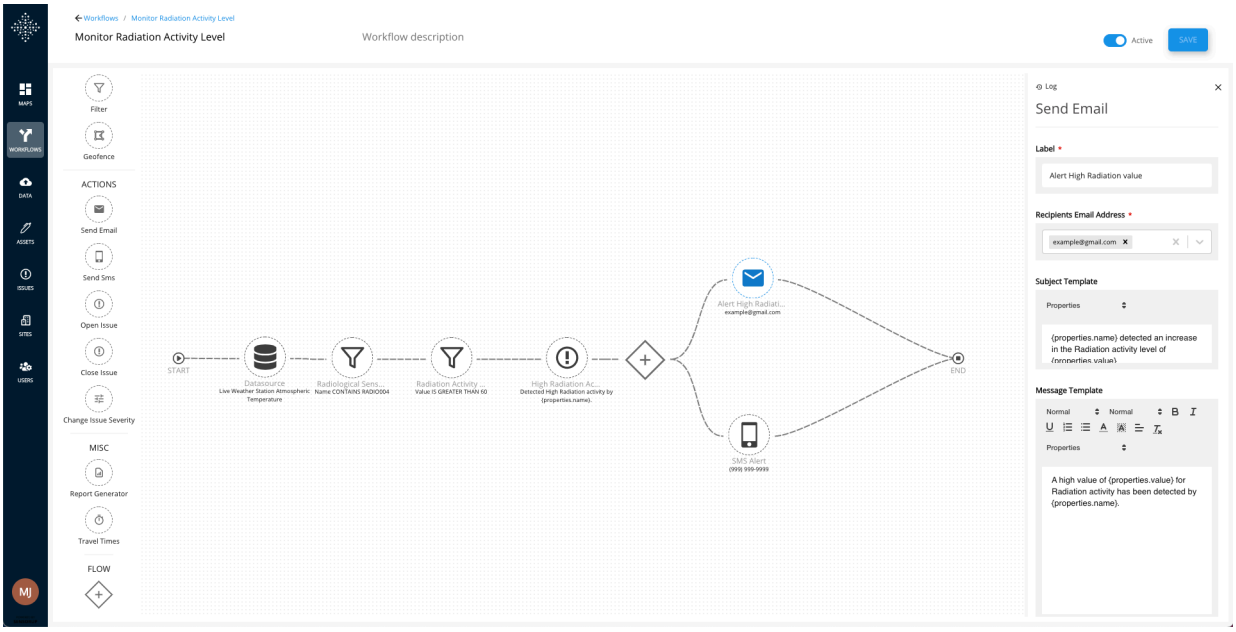

Figure 20 — Client workflow - create alerts

5.1.3. Sensor List

In Testbed 17, the following sensors were used to test the MASBUS STA client instance:

- Two weather stations (an example of this station is provided in Table 3)
- Four biological sensors (an example of this station is provided in Table 4)
- Two chemical sensors (an example of this station is provided in Table 5)
- Four Radiological sensors (an example of this station is provided in Table 6)
- A MISB UAS (Table 7)
- 2 McQ Omnisense Detectors (Table 7)
- MM101 and MM201 Sensors (Table 7)

Table 3 — An Example of a Weather Station Connected to the Client

MULTIDATASTREAM NAME	URLS	
Weather Station ATM001 — Electrical Status	Datastream	/sta/v1.0/MultiDatastreams(22689540763671505)
	MQTT Topic	/sta/v1.0/MultiDatastreams(22689540763671505)/Observations
	ObservedProperties	[dc_voltage_load, dc_current_load, reserve_capacity, timetoempty]
Weather Station ATM001 — Atmospheric Humidity	Datastream	/sta/v1.0/MultiDatastreams(8992411663170205)
	MQTT Topic	/sta/v1.0/MultiDatastreams(8992411663170205)/Observations
	ObservedProperties	[value, error]
Weather Station ATM001 — Atmospheric Precipitation	Datastream	/sta/v1.0/MultiDatastreams(4435531523282605)
	MQTT Topic	/sta/v1.0/MultiDatastreams(4435531523282605)/Observations
	ObservedProperties	[category, rate]
Weather Station ATM001 — Atmospheric Pressure	Datastream	/sta/v1.0/MultiDatastreams(10569985969790905)
	MQTT Topic	/sta/v1.0/MultiDatastreams(10569985969790905)/Observations
	ObservedProperties	[value, error]
Weather Station ATM001 — Link State	Datastream	/sta/v1.0/MultiDatastreams(6255063645059105)
	MQTT Topic	/sta/v1.0/MultiDatastreams(6255063645059105)/Observations
	ObservedProperties	[link_loss, link_state, transmission_range, receive_power, signal_strength, transmit_power]
Weather Station ATM001 — Sensor Location	Datastream	/sta/v1.0/MultiDatastreams(24943357573249205)
	MQTT Topic	/sta/v1.0/MultiDatastreams(24943357573249205)/Observations
	ObservedProperties	[geodetic_latitude, longitude, ellipsoidal_height]
Weather Station ATM001 — Atmospheric Temperature	Datastream	/sta/v1.0/MultiDatastreams(24382037125829805)
	MQTT Topic	/sta/v1.0/MultiDatastreams(24382037125829805)/Observations
	ObservedProperties	[value, error]

MULTIDATASTREAM NAME	URLS	
Weather Station ATM001 — Atmospheric Wind	Datastream	/sta/v1.0/MultiDatastreams(3349516805381705)
	MQTT Topic	/sta/v1.0/MultiDatastreams(3349516805381705)/Observations
	ObservedProperties	[category, speed, direction]

Table 4 — An Example of a Biological Sensor Connected to the Client

MULTIDATASTREAM NAME	URLS	
Biological Sensor BIO001 — Biological Reading	Datastream	/sta/v1.0/MultiDatastreams(10608464675118205)
	MQTT Topic	/sta/v1.0/MultiDatastreams(10608464675118205)/Observations
	ObservedProperties	[material_class, channel_number, particle_density, harmful, bottle_id, integration_time]
Biological Sensor BIO001 — Link State	Datastream	/sta/v1.0/MultiDatastreams(16878926634694905)
	MQTT Topic	/sta/v1.0/MultiDatastreams(16878926634694905)/Observations
	ObservedProperties	[link_loss, link_state, transmission_range, receive_power, signal_strength, transmit_power]
Biological Sensor BIO001 — Sensor Location	Datastream	/sta/v1.0/MultiDatastreams(5523805021019505)
	MQTT Topic	/sta/v1.0/MultiDatastreams(5523805021019505)/Observations
	ObservedProperties	[geodetic_latitude, longitude, ellipsoidal_height]

Table 5 — An Example of a Chemical Sensor Connected to the Client

MULTIDATASTREAM NAME	URLS	
Chemical Sensor CHEM001 — Chemical Reading	Datastream	/sta/v1.0/MultiDatastreams(13050969611475205)
	MQTT Topic	/sta/v1.0/MultiDatastreams(13050969611475205)/Observations
	ObservedProperties	[material_class, material_name, service_number, harmful, chemical_concentration, integration_time, concentration_time, deposition, mass_fraction, gbar_reading, hbar_reading]

MULTIDATASTREAM NAME	URLS	
Chemical Sensor CHEM001 — Link State	Datastream	/sta/v1.0/MultiDatastreams(18485525045055505)
	MQTT Topic	/sta/v1.0/MultiDatastreams(18485525045055505)/Observations
	ObservedProperties	[link_loss, link_state, transmission_range, receive_power, signal_strength, transmit_power]
Chemical Sensor CHEM001 — Sensor Location	Datastream	/sta/v1.0/MultiDatastreams(6849432503413805)
	MQTT Topic	/sta/v1.0/MultiDatastreams(6849432503413805)/Observations
	ObservedProperties	[geodetic_latitude, longitude, ellipsoidal_height]

Table 6 — An Example of a Radiologgical Sensor Connected to the Client

MULTIDATASTREAM NAME	URLS	
Radiological Sensor RADIO001 — Link State	Datastream	/sta/v1.0/MultiDatastreams(11469608031711805)
	MQTT Topic	/sta/v1.0/MultiDatastreams(11469608031711805)/Observations
	ObservedProperties	[link_loss, link_state, transmission_range, receive_power, signal_strength, transmit_power]
Radiological Sensor RADIO001 — Radiological Reading	Datastream	/sta/v1.0/MultiDatastreams(20489400772619305)
	MQTT Topic	/sta/v1.0/MultiDatastreams(20489400772619305)/Observations
	ObservedProperties	[radioactive_material_primary_radiation_code, radioactive_material_category_code, isotope_atomic_number, isotope_mass_number, activity, deposition_concentration, concentration, dose, dose_rate, harmful]
Radiological Sensor RADIO001 — Sensor Location	Datastream	/sta/v1.0/MultiDatastreams(24049175749781705)
	MQTT Topic	/sta/v1.0/MultiDatastreams(24049175749781705)/Observations
	ObservedProperties	[geodetic_latitude, longitude, ellipsoidal_height]

Table 7 — Other Multidatastreams Connected To The Client

MULTIDATASTREAM NAME	URLS	
MISB UAS — Image Frame Geo-Referencing — Geo-Referenced Image Frame	Datastream	/sta/v1.0/MultiDatastreams(25621314725330505)
	MQTT Topic	/sta/v1.0/MultiDatastreams(25621314725330505)/Observations
	ObservedProperties	[geodetic_latitude, longitude, ellipsoidal_height, geodetic_latitude, longitude, ellipsoidal_height, geodetic_latitude, longitude, ellipsoidal_height, geodetic_latitude, longitude, ellipsoidal_height, geodetic_latitude, longitude, ellipsoidal_height]
MISB UAS — Video Moving Target Geo-Referencing — Target Location	Datastream	/sta/v1.0/MultiDatastreams(24897903713488705)
	MQTT Topic	/sta/v1.0/MultiDatastreams(24897903713488705)/Observations
	ObservedProperties	[geodetic_latitude, longitude, ellipsoidal_height]
Predator UAS (MISB) — GeoReferenced Image Frame	Datastream	/sta/v1.0/MultiDatastreams(6960976136389705)
	MQTT Topic	/sta/v1.0/MultiDatastreams(6960976136389705)/Observations
	ObservedProperties	[geodetic_latitude, longitude, ellipsoidal_height, geodetic_latitude, longitude, geodetic_latitude, longitude, geodetic_latitude, longitude, geodetic_latitude, longitude]
Predator UAS (MISB) — Platform Attitude	Datastream	/sta/v1.0/MultiDatastreams(11815127117897505)
	MQTT Topic	/sta/v1.0/MultiDatastreams(11815127117897505)/Observations
	ObservedProperties	[heading_angle, pitch_angle, roll_angle]
Predator UAS (MISB) — Gimbal Attitude	Datastream	/sta/v1.0/MultiDatastreams(6976665876931705)
	MQTT Topic	/sta/v1.0/MultiDatastreams(6976665876931705)/Observations
	ObservedProperties	[yaw_angle, pitch_angle, roll_angle]
Predator UAS (MISB) — Sensor Location	Datastream	/sta/v1.0/MultiDatastreams(6488775276568105)
	MQTT Topic	/sta/v1.0/MultiDatastreams(6488775276568105)/Observations
	ObservedProperties	[geodetic_latitude, longitude, msl_height]
Predator UAS (MISB) — Imaging Sensor Parameters	Datastream	/sta/v1.0/MultiDatastreams(7028895013583705)

MULTIDATASTREAM NAME	URLS	
	MQTT Topic	/sta/v1.0/MultiDatastreams(7028895013583705)/Observations
	ObservedProperties	[sensor_horizontal_fieldof_view, sensor_vertical_fieldof_view]
MCQ Omnisense Detector — radiance	Datastream	/sta/v1.0/MultiDatastreams(18226112642069108)
	MQTT Topic	/sta/v1.0/MultiDatastreams(18226112642069108)/Observations
	ObservedProperties	[detection_code, detail, image, image_resolution, image_compression_ratio, color_image, picture_delay, picture_type, requesting_unit_id, target_box_right1, target_box_left1, target_box_top1, target_box_bottom1, target_direction1, target_box_right2, target_box_left2, target_box_top2, target_box_bottom2, target_direction2, target_box_right3, target_box_left3, target_box_top3, target_box_bottom3, target_direction3, target_box_right4, target_box_left4, target_box_top4, target_box_bottom4, target_direction4, image_id, image_type]
MCQ Omnisense Detector — radiance	Datastream	/sta/v1.0/MultiDatastreams(16767815488006508)
	MQTT Topic	/sta/v1.0/MultiDatastreams(16767815488006508)/Observations
	ObservedProperties	[detection_code, detail, image, image_resolution, image_compression_ratio, color_image, picture_delay, picture_type, requesting_unit_id, target_box_right1, target_box_left1, target_box_top1, target_box_bottom1, target_direction1, target_box_right2, target_box_left2, target_box_top2, target_box_bottom2, target_direction2, target_box_right3, target_box_left3, target_box_top3, target_box_bottom3, target_direction3, target_box_right4, target_box_left4, target_box_top4, target_box_bottom4, target_direction4, image_id, image_type]
MM101-0016-000004-EA—P — taract	Datastream	/sta/v1.0/MultiDatastreams(3808701043167908)
	MQTT Topic	/sta/v1.0/MultiDatastreams(3808701043167908)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM101-0019-000006-EAI-P — taract	Datastream	/sta/v1.0/MultiDatastreams(20255717349771808)
	MQTT Topic	/sta/v1.0/MultiDatastreams(20255717349771808)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM101-0019-000007-EAI-P — taract	Datastream	/sta/v1.0/MultiDatastreams(2548449950950508)
	MQTT Topic	/sta/v1.0/MultiDatastreams(2548449950950508)/Observations

MULTIDATASTREAM NAME	URLS	
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM101-0019-000008-EAM-P — taract	Datastream	/sta/v1.0/MultiDatastreams(13488965271754908)
	MQTT Topic	/sta/v1.0/MultiDatastreams(13488965271754908)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM201-0016-000001-EAM-P — taract	Datastream	/sta/v1.0/MultiDatastreams(15954398545607908)
	MQTT Topic	/sta/v1.0/MultiDatastreams(15954398545607908)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM201-0016-000002-EAI-P — taract	Datastream	/sta/v1.0/MultiDatastreams(25151173847849608)
	MQTT Topic	/sta/v1.0/MultiDatastreams(25151173847849608)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM201-0016-000003-EAI-P — taract	Datastream	/sta/v1.0/MultiDatastreams(11853138511809408)
	MQTT Topic	/sta/v1.0/MultiDatastreams(11853138511809408)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM201-0016-000004-EA—P — taract	Datastream	/sta/v1.0/MultiDatastreams(23407004669847708)
	MQTT Topic	/sta/v1.0/MultiDatastreams(23407004669847708)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]
MM201-0019-000008-EAM-P — taract	Datastream	/sta/v1.0/MultiDatastreams(4259345790791008)
	MQTT Topic	/sta/v1.0/MultiDatastreams(4259345790791008)/Observations
	ObservedProperties	[message_type, detail, amp_hours_remaining]

5.2. MASBUS 3D Client (Botts Inc.)

A second demonstration client was developed to illustrate how MISB metadata extracted from the video feed and converted to standard OGC data feeds can be used to generate a 3D representation of the scene.

The client is built using Cesium JS and OSH Web Client Toolkit. OSH Web Client Toolkit provides a library to connect to OSH server-side APIs (SOS, SensorThings API, SensorWeb API) and supports rendering sensor data using Cesium but also other mapping engines such as OpenLayers, Leaflet, Mapbox, DeckGl. Video data is decoded and displayed using the FFMPEG library compiled to Javascript using Emscripten. The toolkit also provides tools to time-synchronize real-time data streams on the client-side as well as replay historical data.

The following screenshot Figure 21 shows the UAV flying over the access road to the Huntsville, AL arsenal. Platform location and orientation as well as sensor orientation and focal length obtained from the MISB data stream are used to compute the camera view frustum (translucent white).

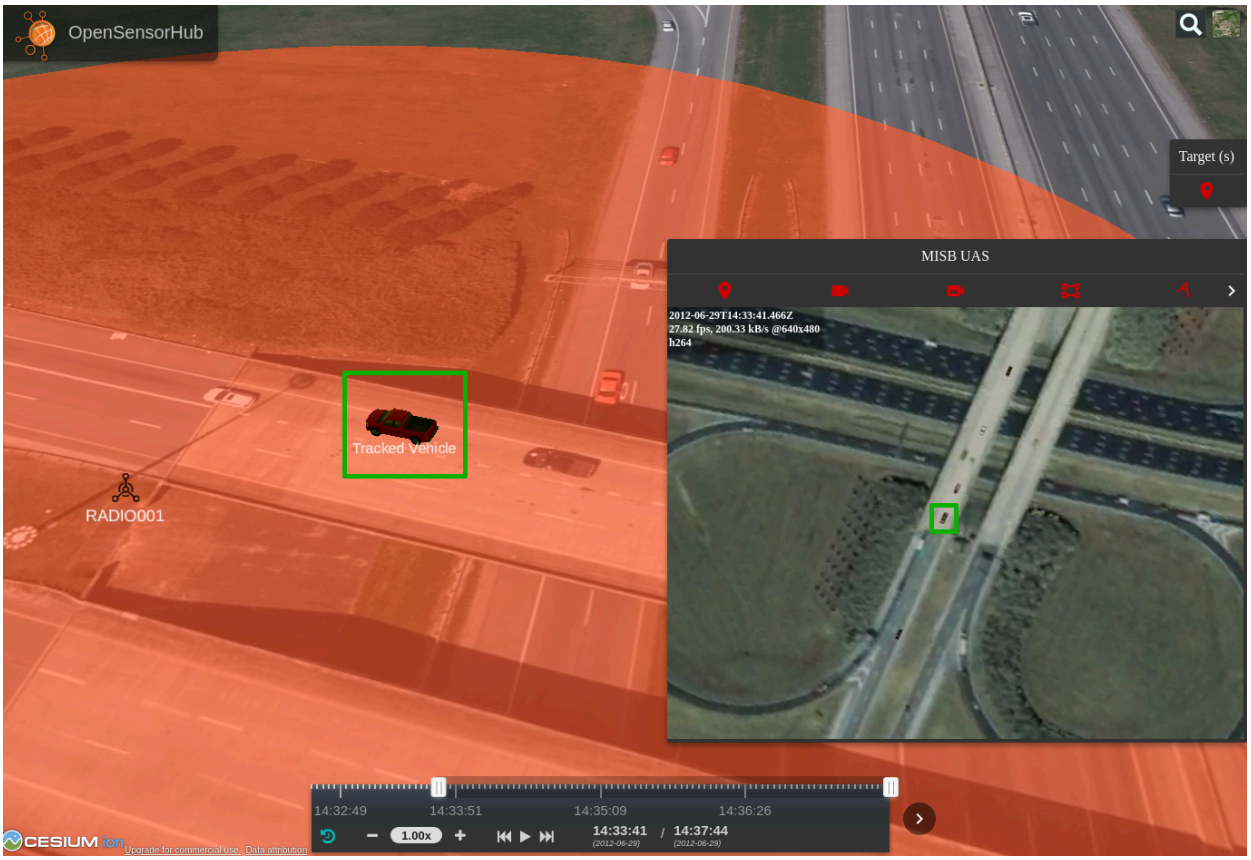

Figure 21 — MASBUS 3D Client showing the UAV flying over

The same MISB metadata is also used to georeference the H264 drone video in real-time. This process called “draping on terrain” is shown on the screenshot below (Figure 22):

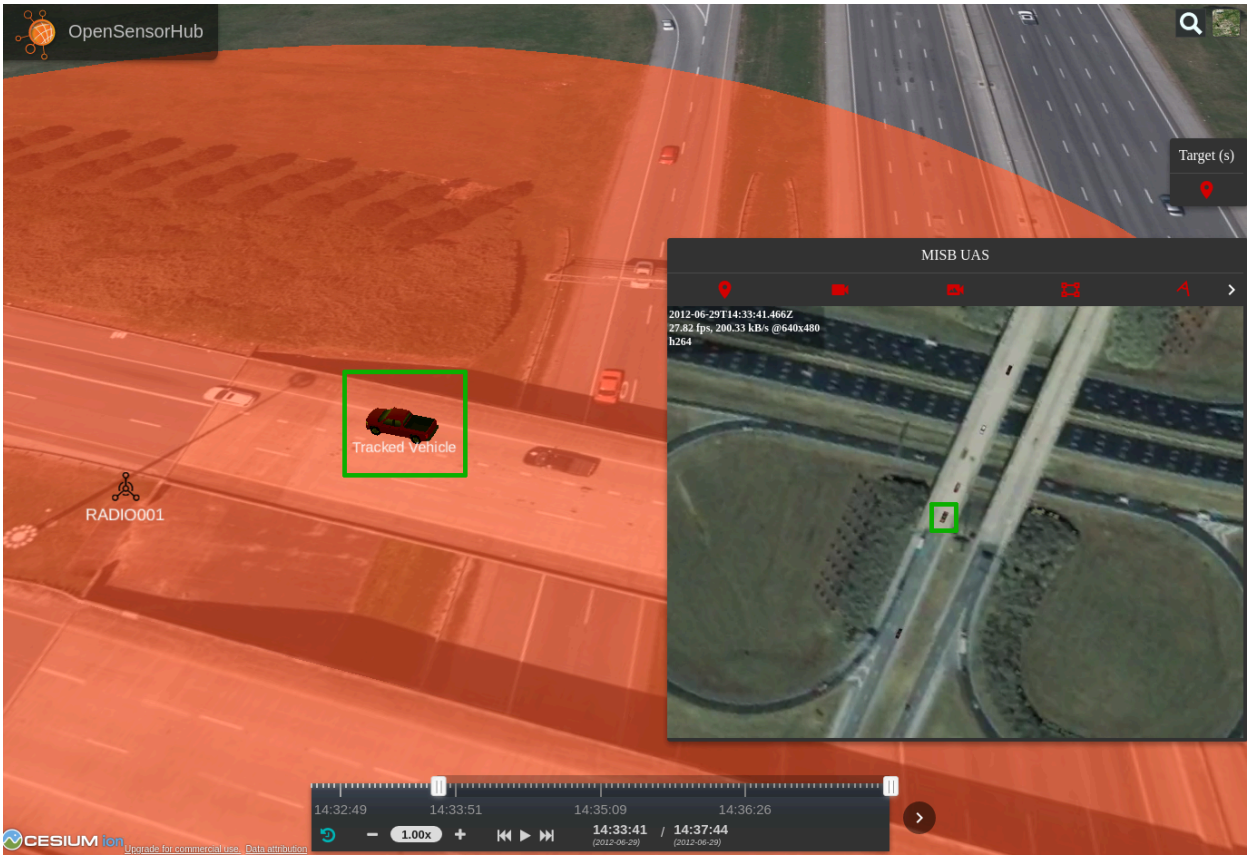

Figure 22 — MASBUS 3D Client- realtime georeferencing

The next screenshot (Figure 23) shows the vehicle identified as carrying radioactive material being tracked automatically by the UAV and triggering the radiation detectors as it passes by (red ellipse).

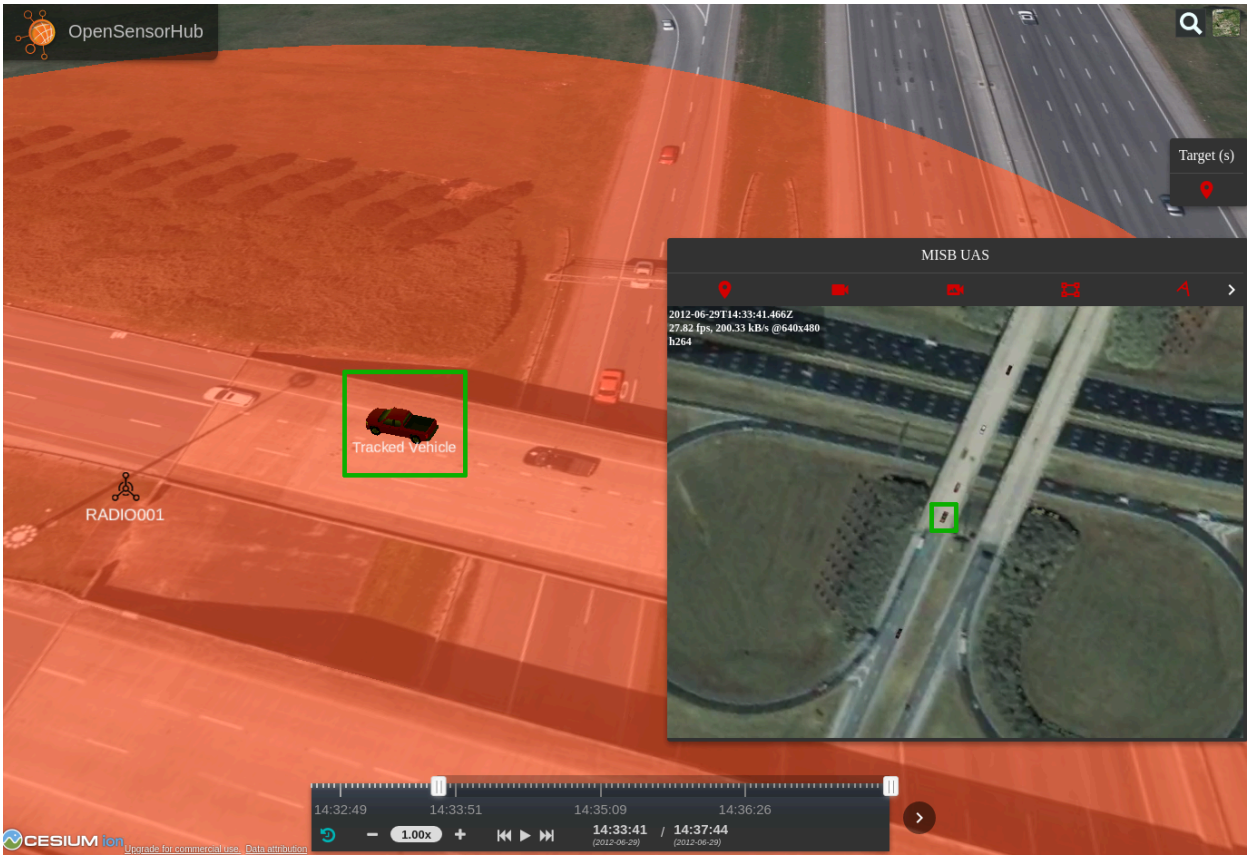

Figure 23 — MASBUS 3D Client- target tracking

The vehicle is detected and tracked in the video (green rectangle in the “MISB UAS” video window) using an image processing algorithm called object tracking. The pixel location of the vehicle is then geolocated (i.e., converted to a lat/lon/alt location) taking into account the drone position, camera optical parameters and terrain. The output of this process is then streamed in real-time using OGC APIs.

See the server implementation section for more details about the target geolocation processing chain.

A video screencast of this client is also available at: https://youtu.be/jcdNYNVDx60

6

EXPERIMENTAL TEST AND RESULTS

6 EXPERIMENTAL TEST AND RESULTS

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. It 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 have been successfully integrated during the Testbed:

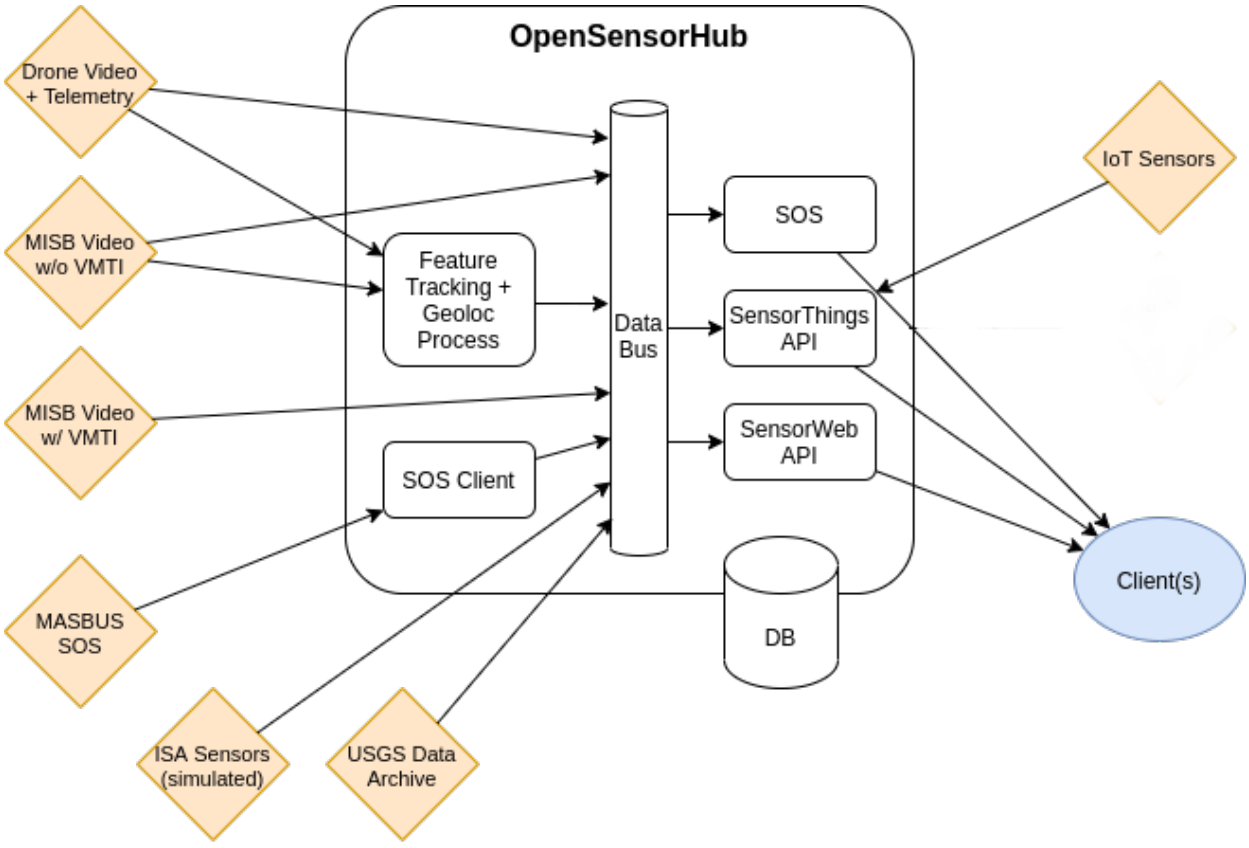

Figure 24 — OSH Based SIF Implementation

6.1. TIE Table

Table 8 — Tie Table

MASBUS SENSOR THINGS SERVER	SENSOR THINGS MASBUS CLIENT
Host: https://ogct17.georobotix.io Port: 1883	https://testbed17-explorer-web-sbx.testbed17-sbx.sensorup.com

Table 9 — Sensor Things MQTT Topics

SENSOR DATA FORMAT	DATASTREAM NAME	STA MQTT TOPIC
	MISB UAS — Image Frame Geo-Referencing — Geo-Referenced Image Frame	/sta/v1.0/MultiDatastreams(25621314725330505)/Observations
	MISB UAS — Video Moving Target Geo-Referencing — Target Location	/sta/v1.0/MultiDatastreams(24897903713488705)/Observations
	Weather Station ATM001 — Electrical Status	/sta/v1.0/MultiDatastreams(22689540763671505)/Observations
	Weather Station ATM001 — Atmospheric Humidity	/sta/v1.0/MultiDatastreams(8992411663170205)/Observations
	Weather Station ATM001 — Atmospheric Precipitation	/sta/v1.0/MultiDatastreams(4435531523282605)/Observations
	Weather Station ATM001 — Atmospheric Pressure	/sta/v1.0/MultiDatastreams(10569985969790905)/Observations
	Weather Station ATM001 — Link State	/sta/v1.0/MultiDatastreams(6255063645059105)/Observations
	Weather Station ATM001 — Sensor Location	/sta/v1.0/MultiDatastreams(24943357573249205)/Observations
	Weather Station ATM001 —	/sta/v1.0/MultiDatastreams(24382037125829805)/Observations

SENSOR DATA FORMAT	DATASTREAM NAME	STA MQTT TOPIC
	Atmospheric Temperature	
	Weather Station ATM001 — Atmospheric Wind	/sta/v1.0/MultiDatastreams(3349516805381705)/Observations
	Weather Station ATM002 — Electrical Status	/sta/v1.0/MultiDatastreams(19513865149297805)/Observations
	Weather Station ATM002 — Atmospheric Humidity	/sta/v1.0/MultiDatastreams(1046064783405805)/Observations
	Weather Station ATM002 — Atmospheric Precipitation	/sta/v1.0/MultiDatastreams(2491691575288905)/Observations
	Weather Station ATM002 — Atmospheric Pressure	/sta/v1.0/MultiDatastreams(19271396263223405)/Observations
	Weather Station ATM002 — Link State	/sta/v1.0/MultiDatastreams(15279632327658905)/Observations
	Weather Station ATM002 — Sensor Location	/sta/v1.0/MultiDatastreams(8899056342873205)/Observations
	Weather Station ATM002 — Atmospheric Temperature	/sta/v1.0/MultiDatastreams(6988155840076705)/Observations
	Weather Station ATM002 — Atmospheric Wind	/sta/v1.0/MultiDatastreams(13449601485073405)/Observations
	Biological Sensor BIO001 — Biological Reading	/sta/v1.0/MultiDatastreams(10608464675118205)/Observations
	Biological Sensor BIO001 — Link State	/sta/v1.0/MultiDatastreams(16878926634694905)/Observations
	Biological Sensor BIO001 — Sensor Location	/sta/v1.0/MultiDatastreams(5523805021019505)/Observations
	Biological Sensor BIO002 — Biological Reading	/sta/v1.0/MultiDatastreams(18093712474454705)/Observations

SENSOR DATA FORMAT	DATASTREAM NAME	STA MQTT TOPIC
	Biological Sensor BIO002 — Link State	/sta/v1.0/MultiDatastreams(21340548997347105)/Observations
	Biological Sensor BIO002 — Sensor Location	/sta/v1.0/MultiDatastreams(5768823060807205)/Observations
	Biological Sensor BIO003 — Biological Reading	/sta/v1.0/MultiDatastreams(25040213394817905)/Observations
	Biological Sensor BIO003 — Link State	/sta/v1.0/MultiDatastreams(5599665382278105)/Observations
	Biological Sensor BIO003 — Sensor Location	/sta/v1.0/MultiDatastreams(9761885450629305)/Observations
	Biological Sensor BIO004 — Biological Reading	/sta/v1.0/MultiDatastreams(10863282023507905)/Observations
	Biological Sensor BIO004 — Link State	/sta/v1.0/MultiDatastreams(17675354851543505)/Observations
	Biological Sensor BIO004 — Sensor Location	/sta/v1.0/MultiDatastreams(19438042852272805)/Observations
	Chemical Sensor CHEM001 — Chemical Reading	/sta/v1.0/MultiDatastreams(13050969611475205)/Observations
	Chemical Sensor CHEM001 — Link State	/sta/v1.0/MultiDatastreams(18485525045055505)/Observations
	Chemical Sensor CHEM001 — Sensor Location	/sta/v1.0/MultiDatastreams(6849432503413805)/Observations
	Chemical Sensor CHEM002 — Chemical Reading	/sta/v1.0/MultiDatastreams(23143666636765505)/Observations
	Chemical Sensor CHEM002 — Link State	/sta/v1.0/MultiDatastreams(11377750814738105)/Observations
	Chemical Sensor CHEM002 — Sensor Location	/sta/v1.0/MultiDatastreams(13676659812286405)/Observations

SENSOR DATA FORMAT	DATASTREAM NAME	STA MQTT TOPIC
	Radiological Sensor RADIO001 — Link State	/sta/v1.0/MultiDatastreams(11469608031711805)/Observations
	Radiological Sensor RADIO001 — Radiological Reading	/sta/v1.0/MultiDatastreams(20489400772619305)/Observations
	Radiological Sensor RADIO001 — Sensor Location	/sta/v1.0/MultiDatastreams(24049175749781705)/Observations
	Radiological Sensor RADIO002 — Link State	/sta/v1.0/MultiDatastreams(16114192159265805)/Observations
	Radiological Sensor RADIO002 — Radiological Reading	/sta/v1.0/MultiDatastreams(25748276702357005)/Observations
	Radiological Sensor RADIO002 — Sensor Location	/sta/v1.0/MultiDatastreams(16196614740126405)/Observations
	Radiological Sensor RADIO003 — Link State	/sta/v1.0/MultiDatastreams(18401542329970105)/Observations
	Radiological Sensor RADIO003 — Radiological Reading	/sta/v1.0/MultiDatastreams(15678579573997305)/Observations
	Radiological Sensor RADIO003 — Sensor Location	/sta/v1.0/MultiDatastreams(7619356560631605)/Observations
	Radiological Sensor RADIO004 — Link State	/sta/v1.0/MultiDatastreams(4469459481100305)/Observations
	Radiological Sensor RADIO004 — Radiological Reading	/sta/v1.0/MultiDatastreams(19264103164103305)/Observations
	Radiological Sensor RADIO004 — Sensor Location	/sta/v1.0/MultiDatastreams(23800596042410705)/Observations
	Predator UAS (MISB) — GeoReferenced Image Frame	/sta/v1.0/MultiDatastreams(6960976136389705)/Observations
	Predator UAS (MISB) — Platform Attitude	/sta/v1.0/MultiDatastreams(11815127117897505)/Observations

SENSOR DATA FORMAT	DATASTREAM NAME	STA MQTT TOPIC
	Predator UAS (MISB) — Gimbal Attitude	/sta/v1.0/MultiDatastreams(6976665876931705)/Observations
	Predator UAS (MISB) — Sensor Location	/sta/v1.0/MultiDatastreams(6488775276568105)/Observations
	Predator UAS (MISB) — Imaging Sensor Parameters	/sta/v1.0/MultiDatastreams(7028895013583705)/Observations
	MCQ Omnisense Detector — radiance	/sta/v1.0/MultiDatastreams(18226112642069108)/Observations
	MCQ Omnisense Detector — radiance	/sta/v1.0/MultiDatastreams(16767815488006508)/Observations
	MM101-0016-000004-EA—P — taract	/sta/v1.0/MultiDatastreams(3808701043167908)/Observations
	MM101-0019-000006-EAI-P — taract	/sta/v1.0/MultiDatastreams(20255717349771808)/Observations
	MM101-0019-000007-EAI-P — taract	/sta/v1.0/MultiDatastreams(2548449950950508)/Observations
	MM101-0019-000008-EAM-P — taract	/sta/v1.0/MultiDatastreams(13488965271754908)/Observations
	MM201-0016-000001-EAM-P — taract	/sta/v1.0/MultiDatastreams(15954398545607908)/Observations
	MM201-0016-000002-EAI-P — taract	/sta/v1.0/MultiDatastreams(25151173847849608)/Observations
	MM201-0016-000003-EAI-P — taract	/sta/v1.0/MultiDatastreams(11853138511809408)/Observations
	MM201-0016-000004-EA—P — taract	/sta/v1.0/MultiDatastreams(23407004669847708)/Observations
	MM201-0019-000008-EAM-P — taract	/sta/v1.0/MultiDatastreams(4259345790791008)/Observations

6.2. Video Demonstration

The client is tested using a simulated data. In the test scenario a target vehicle is driving down a road, passing a couple of Radiological sensors deployed along the road. As the target passes each Radiological sensor, the radiation activity of that sensor rises. Additionally, the target is tracked by a MISB UAS and its corresponding geo-referenced image frames is available through different datastreams and shown on the map (see Figure 25).

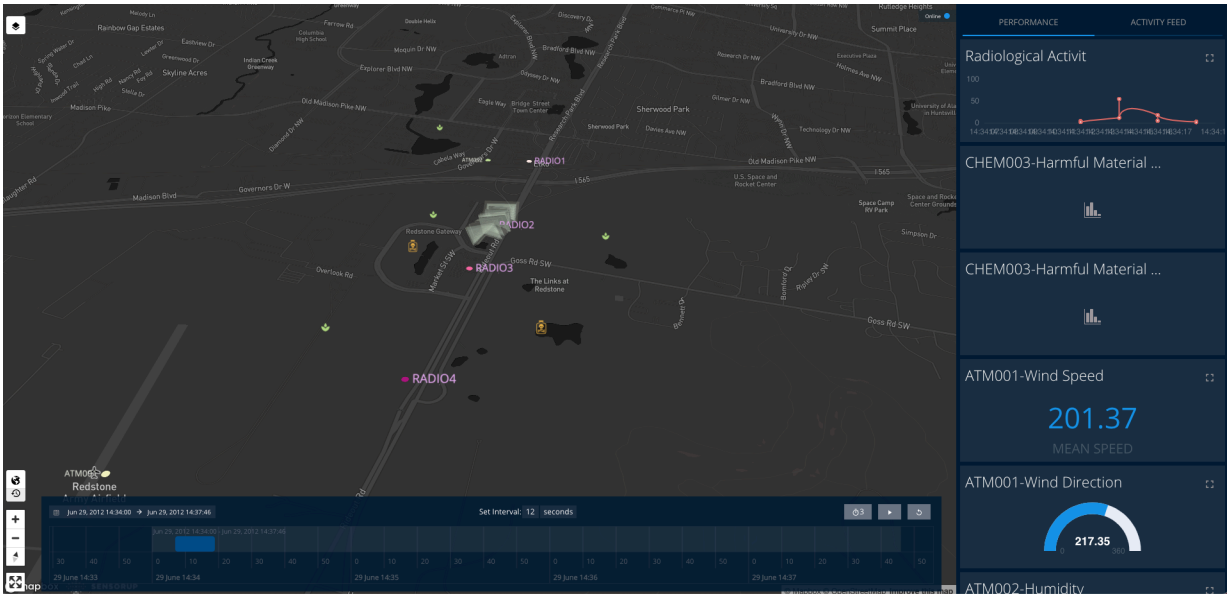

Figure 25 — Testing MASBUS Client (an increase in the radiological activity while the target is passing the sensor RADIO002)

A screen cast of the 2D client is published at: here; and, a screen cast of the 3D client is published at: https://youtu.be/jcdNYNVDx60.

7

FINDINGS

7 FINDINGS

MASBUS integrated implementation involves complex distributed data models and systems with multiple diverse applications. To make well-informed decisions, it is essential to have a proper data integration strategy, which must allow working with heterogeneous data sources and platforms in interoperable ways.

The MASBUS Integration ER describes how interoperability can be established in a common scenario of heterogeneous data sources (*e.g.* sensors, IoT platforms, simulators and other data models and encodings). Moreover, this architecture allows querying and visualizing observations from data sources using widely adopted international standards such as the OGC SensorThings API.

NOTE:

- Interoperability based on standards and on a broad paradigm of observations & measurements.

7.1. Application Development Findings

The following findings were derived from the developed server instance:

- 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).
  - Like all OGC web services, SOS is not well suited for discovery if a large number of offerings are provided (i.e., no standard search capability).

The following findings were derived from the developed server instance:

- The 2D and 3D Dashboard achieved a level of interoperability through implementation of different interfaces to communicate, fetch, infuse and visualize data through REST request/response and MQTT publication-subscription services:
    - Fetching data from the data sources is the first step, building a generic interface which acts as a common platform to fetch data from different data sources providing a common visualizing platform.
    - Automating the process of fetching the live data at multiple intervals.
    - Automating the process of displaying the API response for visualization of data.
    - Injecting the fetched data into different scenario by applying conditions/schemas.

7.2. Recommendations for Future Work

The ER recommends the following activities for further work in future OGC Testbeds and Standards Working Groups:

- Development of the draft Sensor Web API through alignment with both OGC API Common and the OGC SensorThings API. Such alignment could also involve support for the Observations, Measurements, and Samples (OMS) 3.0 candidate standard.
- Advancement of the JSON Encoding Rules SWE Common / SensorML Best Practice (OGC 17-011r2) towards adoption as an OGC Standard.
- Explore the potential use of the Sensor Model Register provided by the OGC Naming Authority (OGC-NA) to deliver semantic definitions in support of MASINT use cases. This could potentially lead to a pan update to the OGC-NA policy on Sensor Models and Parameters (OGC 18-042r4).

A

ANNEX A (INFORMATIVE) REVISION HISTORY

A ANNEX A (INFORMATIVE) REVISION HISTORY

DATE	RELEASE	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
November 15, 2021	.1	Dr. Sara Saeedi	all	initial version
November 18, 2021	.2	OGC Staff	all	document clean up and additional guidance
November 18, 2021	.3	Dr. Carl Reed	various	document clean up and additional guidance
November 19, 2021	1.0	Dr. Sara Saeedi	all	comments integrated
March 29, 2022	1.1	OGC Staff	all	final OGC staff review

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Simon Cox: OGC 10-004r3, *Topic 20 — Observations and Measurements*. Open Geospatial Consortium (2010). https://portal.ogc.org/files/?artifact_id=41579

2. OGC Abstract Specification Topic 20: Geographic information — Observations and Measurements, 2011

3. OGC SensorThings API Part 1: Sensing Version 1.1

4. OData JSON Format Version 4.0 Plus Errata 02

5. H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub: RFC 7946, *The GeoJSON Format*. Internet Engineering Task Force (2016). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.7946.xml

6. OGC SOS (Sensor Observation Service), https://www.ogc.org/standards/sos

7. WebSockets API, https://datatracker.ietf.org/doc/html/rfc6455

8. DDMS 2.0 (Defense Discovery Metadata Standard), https://web.archive.org/web/20110421122454/http://metadata.ces.mil/mdr/irs/DDMS/

9. Semantic Sensor Network (SSN), https://www.w3.org/TR/vocab-ssn/

10. Alexandre Robin: OGC 08-094r1, *OGC® SWE Common Data Model Encoding Standard*. Open Geospatial Consortium (2011). https://portal.ogc.org/files/?artifact_id=41157

11. Mike Botts, Alexandre Robin, Eric Hirschorn: OGC 12-000r2, *OGC SensorML: Model and XML Encoding Standard*. Open Geospatial Consortium (2020). http://docs.ogc.org/is/12-000r2/12-000r2.html

12. Alex Robin: OGC 17-011r2, *JSON Encoding Rules SWE Common / SensorML*. Open Geospatial Consortium (2018). http://docs.opengeospatial.org/bp/17-011r2/17-011r2.html