Open Geospatial Consortium
Submission Date: 2025-03-05
Approval Date: 2025-03-05
Publication Date: 2025-03-05
External identifier of this OGC® document:
http://www.opengis.net/doc/UG/ConnectedSystems-reviewers
Internal reference number of this OGC® document: 23-053r1
Category: OGC® User Guide
Editors: C. Tucker, A. Robin, M. Botts, etc.

OGC API - Connected Systems Standard v1.0 Reviewers Guide

Warning

This document provides guidance for reviewers of the OGC API - Connected Systems Standard v1.0 Candidate. Throughout this document anywhere that there is a reference to the OGC API - Connected Systems Standard v1.0, the reader should understand that until the OGC membership votes to approve the final standard, the OGC API - Connected Systems Standard v1.0 described herein is a Candidate Standard.

This document is a non-normative resource and not an official position of the OGC membership. It is subject to change without notice and may not be referred to as an OGC Standard. In addition to this guide, developers, implementers and reviewers may wish to study the OGC API - Connected Systems Standard v1.0 Reviewers Guide. The guidance provided in this document is not to be referenced as required or mandatory technology in procurements.

Document type:	OGC® User Guide
Document subtype:	
Document stage:	Approved for public release
Document language:	English

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.

Table of Contents

i. Abstract

The OGC API - Connected Systems Reviewers Guide is a public resource structured to provide quick answers to questions which a reviewer may have about the OGC OGC API - Connected Systems Standard v1.0. This OGC document is provided to support professionals who need to understand and/or are reviewing the OGC API - Connected Systems Standard v1.0 but do not wish to implement it.

OGC API - Connected Systems v1.0 is an OGC Implementation Standard for static data (geographic and other domain features) and for dynamic data (e.g., Data Streams: Observations of these Feature Properties, and Control Streams: Commands/actuations that change these Feature Properties) for all manner of Systems (e.g., sensors, things, robots, drones, satellites, control systems, devices, and all manner of Platforms across space, air, land, sea, cyber, and electro-magnetic spectrum).

ii. Keywords

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

Open Geospatial Consortium, OGC API - Connected Systems, ogcdoc, OGC document, OGC Implementation Standard, static data, dynamic data, Data Streams, Control Streams, commands/actuations, sensors, things, robots, drones, satellites, control systems, devices, Platforms, space, air, land, sea, cyber, electro-magnetic spectrum

iii. Preface

This version of the OGC API - Connected Systems Reviewers Guide is limited in scope to the OGC API - Connected Systems v1.0 Standard. Content of this document will be updated when relevant information and feedback to the OGC API - Connected Systems v1.0 Standard is provided and the standard updated. The Open Geospatial Consortium shall not be held responsible for the accuracy or completeness of this reviewers guide.

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.

iv. Submitting organizations

The OGC API - Connected Systems Standards Working Group (SWG) submitted this document for publication by the Open Geospatial Consortium (OGC).

v. Submitters

The OGC API - Connected Systems SWG submitted this document for publication by the OGC.

This page is intentionally left blank.

This page is intentionally left blank.

1. Introduction

1.1. How To Use This Resource

The OGC API - Connected Systems Reviewers Guide is not intended to be read from start to finish. Rather, the document is a resource structured to provide quick answers to questions which a reviewer may have about the OGC API - Connected Systems Standard v1.0. This guide is provided to support professionals who need to understand and/or are reviewing the OGC API - Connected Systems Standard v1.0 but do not wish or have need to implement the Standard.

In addition, this guide can provide insights to professionals considering adopting the OGC API - Connected Systems Standard v1.0 for their projects and products.

The OGC API - Connected Systems Reviewers Guide contains hyperlinks which can be used to navigate directly to relevant sections of the guide as well as to sections of the OGC API - Connected Systems Standard v1.0.

1.2. What is OGC API - Connected Systems Standard v1.0?

The OGC API - Connected Systems Standard v1.0 connects all Systems on or around the Earth into a common 4D framework for the purposes of discovery, access, processing, reasoning, visualization, tasking, and action for all manner of systems (e.g., sensors, things, robots, drones, satellites, control systems, devices, and all manner of Platforms across space, air, land, sea, cyber, and electro-magnetic spectrum), providing a bridge between their dynamic data (e.g., Observations of these Feature Properties, and Commands that change these Feature Properties) and more static representations of them as Features within traditional geographic/geospatial applications.

OGC API - Connected Systems is an OpenAPI/RESTful interface (following the OGC API strategic guidance) that is built upon accepted web formats such as GeoJSON as well as existing OGC information models, including SensorML, Observations and Measurements (O&M) (now called Observations, Measurements and Samples - OMS), SWE Common Data Model, and the Semantic Sensor Network Ontology (SOSA/SSN).

The OGC API - Connected Systems Standard v1.0 is an extension of the OGC API - Features and, in addition to providing its own mechanism for retrieving static and dynamic data from these systems, the API will allow linking to other OGC API Standards, such as OGC API - Maps, OGC API - Coverages, OGC API - Environmental Data Retrieval (EDR), OGC API - 3D GeoVolumes/3D Tiles, SensorThings API (STA), OGC API - Moving Features, OGC API-Processes, and others. (https://ogcapi.ogc.org/connectedsystems/overview.html)

More about the OGC API - Connected Systems Standard v1.0 is available here:

https://ogcapi.ogc.org/connectedsystems/
https://ogcapi.ogc.org/connectedsystems/overview.html
https://github.com/opengeospatial/ogcapi-connected-systems

1.3. Why Is Another Standard Needed?

The development of the OGC API - Connected Systems Standard v1.0 was a response to the OGC's strategic guidance to all Standards Working Groups (SWG) to migrate their legacy/heritage specification baseline to OpenAPI/RESTful patterns. Specifically, the OGC Sensor Web Enablement (SWE) architecture, which has been in global use since 2007, needed to be updated according to this architectural guidance. Also, the evolution of the OGC API-Features, as part of this architectural renaissance, came to offer new opportunities for architectural synergy between the historically divided OGC "web mapping" standards and its "SensorWeb" standards. As mentioned above, the OGC API - Connected Systems Standard v1.0 is an extension of the OGC API - Features, and takes advantage of the modern consensus around other OGC standards such as GeoPose, OMS, Pub/Sub, and more. In the end, this new OGC API - Connected Systems Standard v1.0 represents a modernization and realignment of the OGC's powerful SensorWeb heritage within its new OGC Building Blocks framework.

1.4. How Does OGC API - Connected Systems Standard v1.0 Address Diverse Requirements?

The OGC API - Connected Systems Standard v1.0 addresses a diverse set of requirements from across all domains (e.g., space, air, land, sea, cyber, electro-magnetic spectrum) in a way that bridges these inherently dynamic Systems (whether sensors, things, robots, drones, satellites, control systems, devices, or Platforms) with the more static world of geospatial mapping. The OGC API - Connected System Standard v1.0 supports the discovery, access, processing, reasoning, visualization, tasking, and action for these various dynamic, connected Systems. And, it offers an elegant bridge to other OGC APIs, as outlined above.

1.5. How Was the OGC API - Connected Systems Standard v1.0 Scope Defined?

The scope for the OGC API - Connected Systems Standard v1.0 was very much defined by the OGC's strategic guidance to migrate all legacy/heritage specifications to OpenAPI/RESTful patterns. Once this update was underway, it became apparent that the OGC API - Features refactoring, under this same strategic guidance, offered a unique opportunity for realignment within the larger OGC architecture. The timing of this effort also allowed it to take advantage of recent progress made on other specifications, including those by the GeoPose, OMS, and Pub/Sub SWGs. By aligning all of these different evolutions, the OGC API - Connected Systems Standard v1.0 scope ended up being quite tidily defined.

1.6. Who Will Use the OGC Reviewers Guide?

The OGC API - Connected Systems Reviewers Guide is a resource for those who seek to understand key concepts used in the OGC API - Connected Systems Standard v1.0, the requirements that the standard meets and the data structures the standard specifies.

The OGC intends this guide to be useful for reviewers of the standard as well as decision makers seeking to understand the relevance of this standard in their use cases, and even developers seeking more context.

2. Scope

The OGC API - Connected Systems Reviewers Guide introduces the key concepts used in the OGC API - Connected Systems Standard v1.0 to its target audiences.

To identify broadly applicable requirements for OGC API - Connected Systems Standard v1.0, the SWG solicited use cases and chose to highlight more than a dozen technical use cases and more than half a dozen domain use cases that were agreed to be representative. (See ‘Section 7.0 Use Cases’, below, for more detail). To understand the ways in which the OGC API - Connected Systems Standard v1.0 can be used and how it meets requirements identified, this guide can be used in conjunction with the OGC API - Connected Systems Standard v1.0 use cases section of the standard.

The choices of standardization targets made in the OGC Connected Systems SWG during development of the standard are explained in this section of the present guide.

Finally, this guide explains how the OGC API - Connected Systems Standard v1.0 fits in the landscape of sensors, things (IoT), robotics, drones (e.g., UxS), satellites, control systems, devices, Platforms (of all kinds, across space, air, land, sea, cyber, electromagnetic) and more traditional geospatial computing. The guide explores complementarities between OGC API - Connected Systems Standard v1.0 and approaches that have been taken in other standards for encoding static and dynamic data streams, as well as dynamic control streams, for sensors, things, robots, drones, satellites, control systems, devices, and Platforms of all kinds, across all domains.

The scope of the OGC API - Connected Systems Reviewers Guide can also be defined in terms of what is out of scope. Specifically, the specification definition itself is not within the Reviewer’s Guide. Rather, that info and issues related to its definition are located in OGC GitHub repositories: https://github.com/opengeospatial/ogcapi-connected-systems.

3. Terms and Definitions

The following list is organized alphabetically. The hyperlinks reference many definitions from the OGC/W3C Spatial Data on the Web Working Group's SOSA/SSN Ontology, the OGC's specification definitions, and other such web resources.

***

Actuator: A device that is used by, or implements, an (Actuation) Procedure that changes the state of the world. (https://www.w3.org/TR/vocab-ssn/#SOSAActuator)

Application Programming Interface (API): a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.

Collection: A geospatial resource that may be available as one or more sub-resource distributions that conform to one or more OGC API standards. (https://portal.ogc.org/files/99149)

Command: Command carries the information required by a System to change the state of a Feature of Interest, which may be a System itself, a Subsystem of various Subtypes (e.g, Sensor, Process, Actuator, Platform, Sampler, etc.), or any other Feature. See section *4.2.4.8 Command* below for further definition of this term.

Control Stream: Control Stream defines the channels available for sending Commands to a given System. Among other things, Control Streams provides schemas for the parameters for Commands within the Control Stream. See section *4.2.4.7 Control Stream* below for further definition of this term.

Data Stream: Data Stream is a particular type of (SOSA) ObservationCollection, with the restriction that all Observations are coming from a single System. That is why a DataStream is attached to a System, as a sub resource. See section *4.2.4.5 Data Stream* below for further definition of this term.

Deployment: Describes the Deployment of one or more Systems for a particular purpose. Deployment may be done on a Platform. (https://www.w3.org/TR/vocab-ssn/#SSNDeployment)

Feature: Abstraction of real world phenomena. A digital representation of a real world entity or an abstraction of the real world. Examples of features include almost anything that can be placed in time and space, including desks, buildings, cities, trees, forest stands, ecosystems, delivery vehicles, snow removal routes, oil wells, oil pipelines, oil spill, and so on. The terms feature and object are often used synonymously [ISO-19101]. (https://www.w3.org/TR/sdw-bfp/#dfn-feature, which in turn is referenced by https://docs.ogc.org/is/17-069r4/17-069r4.html#_feature)

Feature Collection: A set of features from a dataset.
(https://docs.ogc.org/is/17-069r4/17-069r4.html)

Feature of Interest: The thing whose property is being estimated or calculated in the course of an Observation to arrive at a Result, or whose property is being manipulated by an Actuator, or which is being sampled or transformed in an act of Sampling.
(https://www.w3.org/TR/vocab-ssn/#SOSAFeatureOfInterest)

GeoPose: GeoPose 1.0 is an OGC Implementation Standard for exchanging the location and orientation of real or virtual geometric objects ("Poses") within reference frames anchored to the earth's surface ("Geo") or within other astronomical coordinate systems. The standard specifies two Basic forms with no configuration options for common use cases, an Advanced form with more flexibility for more complex applications, and five composite GeoPose structures that support time series plus chain and graph structures. (https://www.ogc.org/standard/geopose/)

Implementation Model: For the purposes of the OGC API - Connected Systems standard, we define 'Implementation Model' as the collection of Implementation Standards used to implement the abstract models from the SOSA/SSN Ontology underpinning the design of the OGC API - Connected Systems standard.

Implementation Standards: As the OGC API - Connected Systems standard is an OGC standard, it is considered an Implementation Standard, based on the OGC definition. Within the OGC: "Implementation Standards are different from the Abstract Specification. They are written for a more technical audience and detail the interface structure between software components. An interface specification is considered to be at the implementation level of detail if, when implemented by two different software engineers in ignorance of each other, the resulting components plug and play with each other at that interface."

Observation: Act of carrying out an (Observation) Procedure to estimate or calculate a value of a property of a FeatureOfInterest. Links to a Sensor to describe what made the Observation and how; links to an ObservableProperty to describe what the result is an estimate of, and to a FeatureOfInterest to detail what that property was associated with.
(https://www.w3.org/TR/vocab-ssn/#SOSAObservation)

ObservationCollection: A set of Observations that share at least one common attribute
(https://www.w3.org/TR/vocab-ssn-ext/#sosa:ObservationCollection)

Observations and Measurements: This OGC standard specifies an XML implementation for the OGC and ISO Observations and Measurements (O&M) conceptual model (OGC Observations and Measurements v2.0 also published as ISO/DIS 19156), including a schema for Sampling Features. This encoding is an essential dependency for the OGC Sensor Observation Service (SOS) Interface Standard. More specifically, this standard defines XML schemas for observations, and for features involved in sampling when making observations.

These provide document models for the exchange of information describing observation acts and their results, both within and between different scientific and technical communities. See below. (https://www.ogc.org/standard/om/)

Observations, Measurements and Samples (OMS): This OGC standard builds upon the previous Observations and Measurements standard, and its sister specification in ISO is Topic 20. (https://www.iso.org/standard/82463.html)

Open Geospatial Consortium (OGC): For more than 28 years, Open Geospatial Consortium (OGC) has operated as a neutral forum where government, industry, nonprofits, and academia come together to engage in collective problem-solving around the critical issues of the day. As the global leader in location solutions and related data, OGC is the largest formal community of geospatial experts with a mission to make location information FAIR – Findable, Accessible, Interoperable, and Reusable – for an inclusive and sustainable future. (https://www.ogc.org/)

OGC API: OGC API - Common is a multi-part standard that documents the set of common practices and shared requirements that have emerged from the development of Resource Oriented Architectures and Web APIs within the OGC. Standards developers will use these building-blocks in the construction of other OGC Standards that relate to Web APIs. The result is a modular suite of coherent API standards which can be adapted by a system designer for the unique requirements of their system. As such, this OGC API Standard serves as the "OWS Common" standard for resource-oriented OGC APIs. Consistent with the architecture of the Web, this specification uses a resource architecture that conforms to principles of Representational State Transfer (REST). This OGC API Standard establishes a common pattern that is based on OpenAPI. (https://ogcapi.ogc.org/)

OGC API - Connected Systems: OGC API - Connected Systems v1.0 is an OGC Implementation Standard for connecting all Systems on or around a celestial body such as Earth into a common 4D space for the purposes of discovery, access, processing, reasoning, visualization and tasking. OGC API - Connected Systems v1.0 is built in alignment with OGC API (see above) strategic guidance, as well as well accepted web formats such as GeoJSON as well as existing OGC information models, including SensorML, Observations and Measurements (O&M) (now called Observations, Measurements and Samples - OMS), SWE Common Data Model, and the Semantic Sensor Network Ontology (SOSA/SSN). The OGC API Connected Systems standard is intended to act as a bridge between static data (geographic and other domain Features) and dynamic data (Observations of these Feature Properties, and Commands that change these Feature Properties). (https://ogcapi.ogc.org/connectedsystems/)

OGC API - Features: OGC API - Features is a multi-part standard that offers the capability to create, modify, and query spatial data on the Web and specifies requirements and recommendations for APIs that want to follow a standard way of sharing feature data. The specification is a multi-part document. The Core part of the specification describes the mandatory capabilities that every implementing service has to support and is restricted to read-access to spatial data. Additional capabilities that address specific needs will be specified in additional parts. Envisaged future capabilities include, for example, support for creating and modifying data, more complex data models, richer queries, and additional coordinate reference systems. (https://ogcapi.ogc.org/features/)

OGC API - Pub/Sub: The OGC API - Connected Systems specification implements the Pub/Sub mechanism in OGC API - Environmental Data Retrieval - Part 2: Publish-Subscribe Workflow, which has been referred to the newly rechartered OGC Pub/Sub SWG for formal consideration and passage (https://github.com/opengeospatial/pubsub).

OpenAPI: The OpenAPI Specification is a specification language for HTTP APIs that provides a standardized means to define your API to others. You can quickly discover how an API works, configure infrastructure, generate client code, and create test cases for your APIs. Read more about how you can get control of your APIs now, understand the full API lifecycle and communicate with developer communities inside and outside your organization. (https://www.openapis.org/)

Platform: A Platform is an entity that hosts other entities, particularly Sensors, Actuators, Samplers, and other Platforms. (NOTE: Within SOSA/SSN, a Platform is not a System, but within the OGC API - Connected System specification, a System Resource can implement both SSN System and SOSA Platform classes.). (https://www.w3.org/TR/vocab-ssn/#SOSAPlatform)

Procedure: A workflow, protocol, plan, algorithm, or computational method specifying how to make an Observation, create a Sample, or make a change to the state of the world (via an Actuator). A Procedure is re-usable, and might be involved in many Observations, Samplings, or Actuations. It explains the steps to be carried out to arrive at reproducible Results. (NOTE: A Procedure can describe a particular “make and model” of a System (as in a ‘Data Sheet’), or a list of steps a human does.) (https://www.w3.org/TR/vocab-ssn/#SOSAProcedure)

Process: The Process concept is not explicitly defined in SOSA/SSN. Rather, depending on the type of processing algorithm, a Process is just a regular System tagged using one of the sub types defined previously. See section *4.2.4.1.1.5 Process* below for further definition of this term.

REST: Representational state transfer (REST) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web. REST defines a set of constraints for how the architecture of an Internet-scale distributed hypermedia system, such as the Web, should behave. The REST architectural style emphasizes the scalability of interactions between components, uniform interfaces, independent deployment of components, and the creation of a layered architecture to facilitate caching of components to reduce user-perceived latency, enforce security, and encapsulate legacy systems. The term REST was first coined by Roy Thomas Fielding in 2000. (Fielding, Roy Thomas (2000). "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.). University of California, Irvine.)

REST has been employed throughout the software industry and is a widely accepted set of guidelines for creating stateless, reliable web APIs. A web API that obeys the REST constraints is informally described as RESTful. RESTful web APIs are typically loosely based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data. (Note: OGC API - Connected Systems Standard v1.0 uses JSON).

The REST architecture makes use of four commonly used HTTP methods. These are:

Method	Description
GET	This method helps in offering read-only access for the resources.
POST	This method is implemented for creating a new resource.
DELETE	This method is implemented for removing a resource.
PUT	This method is implemented for updating an existing resource or creating a fresh one.

Sampler: A device that is used by, or implements, a (Sampling) Procedure to create or transform one or more samples. (https://www.w3.org/TR/vocab-ssn/#SOSASampler)

Sampling Feature: A Feature, such as a station, transect, section or specimen, which is involved in making observations concerning a domain feature. Sampling Features are artifacts of an observational strategy, and have no significant function outside of their role in the observation process. (OGC and ISO 19156:2011(E)) (https://www.ogc.org/standard/om/)

For the purposes of the OGC API - Connected Systems Standard, additional clarification is useful.

Sampling Features describe exactly what part of a real-world feature (e.g. domain feature) is observed or controlled. Sampling features can be physical (e.g. Specimen, MaterialSample), or more abstract; for example statistical (e.g. StatisticalSample), or simply a geometric shape (e.g. SamplingPoint, SamplingCurve for lines and profiles, or volumes like SamplingSphere and SamplingFrustum).

Sensor: Device, agent (including humans), or software (simulation) involved in, or implementing, a Procedure. Sensors respond to a Stimulus, e.g., a change in the environment, or Input data composed from the Results of prior Observations, and generate a Result. Sensors can be hosted by Platforms. (https://www.w3.org/TR/vocab-ssn/#SOSASensor)

Sensor Model Language (SensorML): SensorML is an OGC standard that provides a robust and semantically-tied means of defining processes and processing components associated with the measurement and post-measurement transformation of observations. This includes sensors and actuators as well as computational processes applied pre- and post-measurement. The main objective is to enable interoperability, first at the syntactic level and later at the semantic level (by using ontologies and semantic mediation), so that sensors and processes can be better understood by machines, utilized automatically in complex workflows, and easily shared between intelligent sensor web nodes. This standard is one of several implementation standards produced under OGC's Sensor Web Enablement (SWE) activity. SensorML 3.0 is a standalone part of the OGC API - Connected Systems suite of specifications. (https://www.ogc.org/standard/sensorml/)

SOSA/SSN: The W3C Semantic Sensor Network Incubator Group ontology (SSN), later revised by the OGC/W3C Spatial Data on the Web Working Group (SDWWG), and expanded based on the Sensor, Observation, Sample, and Actuator (SOSA) ontology. Similar to the original SSO, SOSA acts as a central building block for the SSN but puts more emphasis on light-weight use and the ability to be used standalone. (https://www.w3.org/TR/vocab-ssn/). Note: A RDFS/OWL implementation of SOSA/SSN exists, but is not used in this specification since the specification team decided to favor Feature encodings like GeoJSON, JSON-FG, etc.

SWE Common Data Model Encoding Standard: The Sensor Web Enablement (SWE) Common Data Model Encoding Standard (heretofore "OGC SWE Common") defines low level data models for exchanging sensor related data between nodes of the OGC® Sensor Web Enablement (SWE) framework. These models allow applications and/or servers to structure, encode and transmit sensor datasets in a self describing and semantically enabled way. SWE Common 3.0 is a standalone part of the OGC API - Connected Systems Standard suite of specifications. https://www.ogc.org/standard/swecommon/

Subsystem: A component of a parent System (subsystem is just a property of the SSN model, because a Subsystem is just a System within a System. (https://www.w3.org/TR/vocab-ssn/#SSNhasSubsystem)

System: System is a unit of abstraction for pieces of infrastructure that implement Procedures. A System may have components, its Subsystems, which are other Systems. (https://www.w3.org/TR/vocab-ssn/#SSNSystem)

4. Conceptual Overview

In this conceptual overview, we address the information model at the core of the OGC API - Connected System Standard v1.0, as well as the API design.

4.1 Information Model

The OGC API - Connected Systems Standard v1.0 is built upon an information model with two parts - the conceptual models and implementation models. The latter are based on the former. All have a deep history within the Open Geospatial Consortium and World Wide Web Consortium processes.

4.1.1 Conceptual Model

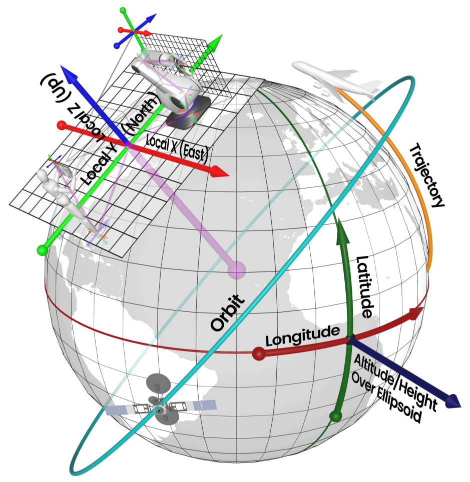
The conceptual model underpinning the OGC API - Connected Systems Standard v1.0 has two major components. First is the joint OGC/W3C Spatial Data on the Web Working Group's (SDWWG) SOSA/SSN model, and the SOSA/SSN revisions underway based on the inclusion of Observation, Measurement and Sampling (OMS) updates to the OGC Observations and Measurement (O&M) model. Together, these provide a conceptual model for describing every possible System, their sensing, processing, and actuating Subsystems, and the Data Streams and Control Streams at their core. Second is the GeoPose standard from the OGC which provides a conceptual model for describing a digital object's pose defined relative to a geographical frame of reference. This allows for the description of the information required to anchor a particular System, and its various Subsystems, in space and time with the kinds of rigorous positional (e.g., spatio-temporal and orientation) information required to sense and act with geographic precision and accuracy.

4.1.1.1 SOSA/SSN/OMS

SOSA/SSN has a long history. The OGC/W3C SDWWG did outstanding work bringing together a diverse community of thought leaders and practitioners from academia, industry and government to assemble a standard ontology for sensor semantics. The latest updates have their history in SOSA/SSN concepts (particularly Observer and Deployment classes) that were not originally included in the Observations and Measurements (O&M) standard. The OGC O&M community saw their value, and this inspired the evolution toward the Observations, Measurements and Samples (OMS) standard. In turn, the SOSA/SSN community saw value in updating SOSA/SSN to reflect the OGC OMS efforts. As a result, these SOSA/SSN revisions represent core learning within the SOSA/SSN community that advances the SOSA/SSN without breaking backward compatibility. The OGC API - Connected System Standard v1.0 is deliberately built on this OMS update of SOSA/SSN, using SensorML as its Implementation Model.

4.1.1.2 GeoPose

The OGC GeoPose standard builds on decades of experience defining the position of objects in geographic space and time. Conceptually, when a real or digital object's pose is defined relative to a geographical frame of reference it will be called a "geographically-anchored pose." All physical world objects inherently have a geographically-anchored pose. Digital objects may be associated with a geographically-anchored pose (for example, in a real-world overlay or on a stage). Specifically, the OGC GeoPose standard defines the rules for the interoperable interchange of geographically-anchored poses. As such, the OGC GeoPose standard defines a conceptual model, a logical model, and encodings for the position and orientation of a real or a digital object



in machine-readable forms using real world coordinates. (For more on GeoPose, read the OGC GeoPose Reviewers Guide, https://docs.ogc.org/guides/22-000.html)

4.1.2 Implementation Model

The Implementation Model underpinning the OGC API - Connected Systems Standard v1.0 has three major components. First is Sensor Model Language (SensorML), which is the OGC standard for encoding descriptions of sensing Systems and their Observations, as well as Processes, Procedures and Deployments. Second is the OGC SWE Common Data Encoding Standard which allows for the detailed common/standard way of describing the schemas of any Data Stream, regardless of its format, including binary formats. Third is JSON and binary methods for encoding Observations, beyond the traditional XML encodings. The first two of these Implementation Models have been at the core of the OGC SWE legacy architecture, and they continue as core concepts within the OGC API - Connected Systems standard. The third is based on OGC Best practices for JSON encodings of SensorML and OGC SWE Common that have been developed over time (https://docs.ogc.org/bp/17-011r2/17-011r2.html). Work has also been done on binary encodings such as Protobuf and FlatGeobuf. Together, these Implementation Models serve as the core of the new OGC API - Connected Systems Standard v1.0.

4.1.2.1 SensorML

The primary focus of the Sensor Model Language (SensorML) is to provide a robust and semantically-tied means of defining Processes and processing components associated with the measurement and post-measurement transformation of Observations. This includes Sensors and Actuators as well as computational Processes applied pre- and post-measurement. The main objective is to enable interoperability, first at the syntactic level and later at the semantic level (by using ontologies and semantic mediation), so that Sensors and Processes can be better understood by machines, utilized automatically in complex workflows, and easily shared between intelligent SensorWeb nodes. This standard is one of several implementation standards produced under OGC's Sensor Web Enablement (SWE) activity. There has long been an OGC Best Practice for a SensorML JSON encoding (https://docs.ogc.org/bp/17-011r2/17-011r2.html) that is being formalized as SensorML 3.0 (https://docs.ogc.org/DRAFTS/23-000.html), a standalone part of the OGC API - Connected Systems Standard v1.0 suite of specifications.

4.1.2.3 SWE Common

The Sensor Web Enablement (SWE) Common Data Model Encoding Standard defines low level data models for exchanging sensor related data between nodes of the OGC® Sensor Web Enablement (SWE) framework. These models allow applications and/or servers to structure, encode and transmit sensor datasets in a self describing and semantically enabled way.

There has long been an OGC Best Practice for SWE Common Data Model JSON encoding (https://docs.ogc.org/bp/17-011r2/17-011r2.html) that is being formalized as SWE Common Data

Model 3.0 (https://docs.ogc.org/DRAFTS/24-014.html), a standalone part of the OGC API - Connected Systems Standard v1.0 release.

There is complementary work being done within the OGC API - Features SWG that could provide a parallel implementation that performs the same functions as SWE Common. A complete alignment with the OGC API - Features future feature schema handling would be done in future versions of OGC API - Connected Systems Standard v1.0 suite of specifications.

4.1.2.4 Observation JSON/Binary Encodings

The OGC API - Connected Systems Standard v1.0 suite of specifications includes JSON encodings for Observations. A future Part 5 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use these binary formats (e.g. Protobuf, Flatbuf, Apache Avro™, etc.).

Note: FlatGeoBuf does have limitations as to its applicability to Observations (see 4.1.2.4 above), particularly with regard to video and other high bandwidth data types. This is why Flatbuf is referenced here rather than FlatGeoBuf, which is used to encode Features and Geometries (see below).

4.1.2.5. Features and Geometries JSON (JSON-FG)/Binary Encodings

JSON-FG is used as one possible encoding for Feature Resources (e.g., System, Procedure, etc.). There is ongoing work within the OGC API - Feature SWG to define Protobuf and FlatGeobuf encodings of Features and Geometries.

4.2 API Design

The design of the OGC API - Connected Systems Standard v1.0 can best be understood in terms of the overarching strategic guidance that inspired it, the complementary OGC API standards that it is extending or reusing, and then the structure of the API design itself.

4.2.1 OGC API Strategic Guidance

OGC API strategic guidance has led OGC SWGs to reimagine their existing specifications in accordance with OpenAPI/RESTful architectural patterns that define reusable API building blocks with responses in JSON and HTML. The resulting OGC API standards define modular API building blocks to spatially enable Web APIs in a consistent way. The OGC API family of standards is organized by resource type. OGC API - Common defines the resources and access mechanisms which are useful for a client seeking to understand the offerings and capabilities of an API. The standard also provides a common connection between the API landing page and resource-specific details.

The OGC API - Connected Systems Standard v1.0 intentionally embraces these OpenAPI/RESTful patterns, particularly following OGC API guidance.

4.2.2 OGC API - Features

More specifically, the OGC API - Connected Systems Standard v1.0 is an extension of the OGC API - Features standard. This decision was made because most of the concepts in the conceptual model, discussed above, are features. The OGC API - Features standard provides a way of encoding Features in multiple formats (including binaries) and the Feature API SWG is working on Part 5 to provide schemas for these encodings. As mentioned above, this creates a future opportunity to harmonize 'SWE Common' and 'Feature Schemas' that the Connected Systems SWG is eager to pursue.

4.2.3 OGC API - Pub/Sub

Both the OGC API - Connected Systems and OGC API - Features SWG's have agreed to align with the Pub/Sub mechanism in Part 2 of the OGC API - Environmental Data Retrieval. This includes using AsyncAPI, and an agreement to generate an MQTT profile. This work stream has been referred over to the previously dormant OGC Pub/Sub SWG. The OGC API - Connected Systems Standard will include a future Part 3 extension that will provide detailed guidance on the use of various Pub/Sub protocols including MQTT, AMQP, DDS, Kafka, etc. (See sections 5.2.2.1.2 and 5.2.3.1.2 and section "6.2 Other Web Standards" for more discussion).

4.2.4 OGC API - Connected Systems Standard v1.0

In the end, the OGC API - Connected Systems Standard v1.0 puts all this together in a way that maximizes re-use and interoperability in a way that allows for the connection of all Systems on planet earth and beyond. All of the resources within the OGC API - Connected Systems Standard v1.0 are based on SOSA/SSN concepts, and encoded in SensorML and various encodings for Observations (e.g., JSON, Protobuf, FlatGeobuf, etc.). Due to the alignment with OGC API - Features, the OGC API - Connected Systems SWG chose to make as many of these resources as possible Feature Resources. Specifically, Feature Resources include Systems (and Subsystems) of various subtypes (e.g., Sensors, Actuators, Platforms, Samplers, Processes), Procedures, Deployments, Sampling Features. Additionally, the OGC API - Connected Systems Standard includes non-Feature Resources - specifically Data Streams, Observations, Control Streams, and Commands. This distinction will be discussed further below. For ease of reading and comprehension, many Terms and Definitions (from section 3 above) will be repeated in this section.

4.2.4.1 System (and Subsystem)

We begin this discussion with the SOSA/SSN definition for System:

> System is a unit of abstraction for pieces of infrastructure that implement Procedures. A System may have components, its Subsystems, which are other Systems.
> https://www.w3.org/TR/vocab-ssn/#SSNSystem

In the real world, Systems will include things that normal people consider sensors, things, robots, drones, satellites, control systems, devices, and Platforms of all kinds, across the domains of space, air, land, sea, cyber, and electro-magnetic spectrum. In truth, all of these Systems represent various constellations of Sensors, Processes and Actuators, designed to accomplish various goals, which can be hierarchically combined in any way to address specific real world problems. In more abstract terms, according to SOSA/SSN, a System is “a unit of abstraction for pieces of infrastructure that implement Procedures. A System may have components, its Subsystems, which are other Systems.”

Thus, we end this discussion with the SOSA/SSN definition of Subsystem, which is the *has Subsystem* characteristic of a System:

> Relation between a System and its component parts.
> https://www.w3.org/TR/vocab-ssn/#SSNhasSubsystem

Many Systems can be Sensors, Actuators and Processes at the same time. In particular, a Sensor can often accept Commands (e.g. change sampling rate or sensitivity), and Actuators can produce data (e.g. Actuator status). It is often cumbersome to create a separate Systems in these cases.[1]

Any system subtype can have Data Streams and Control Streams. For instance, a System Subtype (Sensor) may have Control Streams, and a System Subtype (Actuator) may have Data Streams.

4.2.4.1.1 System Subtype

We begin this discussion by recognizing that the OGC API - Connected Systems Standard v1.0 idea of System Subtype builds on the SOSA/SSN ontology, with some specific additions in order to address the full set of Connected Systems use cases. To foreshadow the coming discussion, these SubType definitions include:

> Sensor
> Actuator
> Platform
> Sampler
> Process

[1] Additionally, a given System might have combinations of different kinds of Subsystem subtypes simultaneously. In this case, each Subsystem would be described as their own type. In the end, this is rather arbitrary, and it is up to the system designer (or the system modeler) to model the system in the way that best works for your use case. At some point, the system modeler creates a black box and says ‘this is what this black box does”. The system designer may have good reason to articulate all of the system capabilities in all of their details while the system modeler may want to keep it simpler for the purposes of their use case.

These Systems (and Subsystems) have various Subtypes (see below), though more complex systems can engender all of these subtypes simultaneously. It is common for lay people to refer to different kinds of systems in different ways as one subtype manifests as the dominant characteristic. For instance, many Systems are thought of primarily as “sensors” or “sensing systems” even though they have processes and actuators within them. Other Systems are thought of primarily as “processes” or “processors” even though they have Sensors and Actuators packaged within them. Other Systems are thought of primarily as “actuators,” even though they have Sensors and onboard processing that make them work. It is often the case that a given System is, quite simply, complex - such as an aircraft, or a satellite, or a control system, with many different Subsystems of different Subtypes integrated for a very specific purpose. And, the position/orientation (e.g., GeoPose) of each of these Subsystems can be different, depending on how they are mounted and operated on the larger Platform at the heart of the System. Due to this complexity, it is critical that these various System Subtypes be semantically tagged.

4.2.4.1.1.1 Sensor

We begin this discussion with the SOSA/SSN definition for Sensor:

> “Device, agent (including humans), or software (simulation) involved in, or implementing, a Procedure. Sensors respond to a Stimulus, e.g., a change in the environment, or Input data composed from the Results of prior Observations, and generate a Result. Sensors can be hosted by Platforms.” (https://www.w3.org/TR/vocab-ssn/#SOSASensor)

Again, the term Sensor can be used to describe any sensing system or Subsystem that observes the world and generates Data Streams filled with Observations (see below). In the world of the OGC API - Connected Systems Standard v1.0, we recognize that all such Sensors exist oriented on (or around) Earth at a given moment in time, and therefore should have GeoPose information for every Observation. Of course, not all Sensors are integrated with complementary Sensors required to provide position, navigation and timing (PNT) solutions capable of generating complete GeoPose information. The Sensor System/Subsystem may have a magnetic compass that provides directionality, but no source for location information, such as GPS. Others may offer location and direction, but lack the accelerometers required to derive orientation. At an integration level, there tend to be engineering methods that allow for the field augmentation of a given Sensor with the requisite Sensors, so that GeoPose information can be provided for all Observations.

Even this simple example of ensuring telemetry Sensors are properly paired with the “primary’ Sensor demonstrates some of the complexities associated with properly incorporating Sensors into a common 4D framework via the OGC API - Connected Systems Standard.

Within the OGC API - Connected Systems Standard v1.0, Sensor Observations are shared over Data Streams, within which Observations are sent. (See below). The control of Sensors (which are Systems) is done with Control Streams (See below).

4.2.4.1.1.2 Actuator

We begin this discussion with the SOSA/SSN definition for Actuator:

> "A device that is used by, or implements, an (Actuation) Procedure that changes the state of the world." (https://www.w3.org/TR/vocab-ssn/#SOSAActuator)

Again, it is important to understand that an Actuator can be a kind of System with other System Subtypes as Subsystems. Within the OGC API - Connected Systems Standard v1.0, Actuators can be as simple or as complex as needed. This could be something as simple as a door lock Actuator which has a Sensor on it that confirms the status of the door lock (e.g., locked, unlocked), and an onboard Process which sends an alert to the physical security System. It could also be quite complex, involving the control of a gimbal, the tasking/control/dispatch of a drone, the tasking of a satellite, or the launching of a countermeasure.

Within the OGC API - Connected Systems Standard v1.0, Actuators are controlled by Control Streams, within which Commands are sent. (See below).

4.2.4.1.1.3 Platform

We begin this discussion with the SOSA/SSN definition of Platform:

> "A Platform is an entity that hosts other entities, particularly Sensors, Actuators, Samplers, and other Platforms." (https://www.w3.org/TR/vocab-ssn/#SOSAPlatform).

The OGC API - Connected Systems Standard v1.0, and the underlying SensorML specification, provide an additional use case not contemplated within the SOSA/SSN ontology. Specifically, a Platform can also be a System if you combine both Platform and System class from SSN together. In the OGC API - Connected Systems Standard v1.0, all Platforms are also Systems.

4.2.4.1.1.4 Sampler

We begin this discussion with the SOSA/SSN definition of Sampler:

> "A device that is used by, or implements, a (Sampling) Procedure to create or transform one or more samples." (https://www.w3.org/TR/vocab-ssn/#SOSASampler)

Sometimes the distinction between the Sampler and the Sensor is not evident, as they are often packaged as a unit. The same device may be a Sampler when it is used to take a Sample, but a Sensor when it is deployed as a sensing System that is systematically collecting Data Streams filled with Observations.

Also, a Sampler need not be a physical device. It could be a person collecting Samples via a Procedure.

The concept of a Sampler is useful when the sampling methodology needs to be documented separately from the Sensor that actually makes the measurement. This is often the case when the measurement is made ex-situ (e.g. a blood sample collected by a nurse and later analyzed in the lab), or when a chain of samples is involved (e.g. a rock core sample collected in the field is broken down into smaller segments that are then analyzed with various instruments).

4.2.4.1.1.5 Process

We begin this discussion with the OGC API - Connected Systems Standard v1.0 definition of Process, since there is no such SOSA/SSN definition:

> The Process concept is not explicitly defined in SOSA/SSN. Rather, depending on the type of processing algorithm, a Process is just a regular System tagged using one of the sub types defined previously.

A Process would thus be classified as:

- A Sensor if the process simulates observations or acts on input observations to generate derived observations ;
- An Actuator if the process computes lower level actuations from a higher level command ;
- A Platform if the Process simulates a moving Platform ;
- A Sampler if the Process simulates a sampling Procedure.
- etc.

However, in the OGC API - Connected Systems Standard v1.0, a second property is available to describe the type of asset that is involved in the implementation of the System. This property called "assetType" can take the value "process" or "simulation".

Note that a Process instance is different from Procedure. The Procedure is what describes the implementation and characteristics of any System, including processes but also hardware equipment or even human behavior (see below).

4.2.4.2 Procedure

We begin this discussion with the SOSA/SSN definition of Procedure:

> “A workflow, protocol, plan, algorithm, or computational method specifying how to make an Observation, create a Sample, or make a change to the state of the world (via an Actuator). A Procedure is re-usable, and might be involved in many Observations, Samplings, or Actuations. It explains the steps to be carried out to arrive at reproducible Results.”

Within SOSA/SSN, a System ***implements*** a Procedure. For a piece of equipment, Procedures represent ***types*** of Systems, Sensors, Processes, Actuators, Platforms and Samplers and the procedure description usually corresponds to the system's datasheet. But in the case where a System (or Platform) involves one or more persons, (referred to in SOSA/SSN as an ‘Agent, including Humans’), the Procedure description would describe the methodology used by these persons.

Note that when a given System is capable of implementing different Procedures, the OGC API - Connected Systems Standard v1.0 provides several ways to describe this:

- As a single System instance associated with a Procedure with multiple "modes" (see SensorML Modes) if those are known in advance.
- As multiple System instances referring to the same "person" in the contact information.

Also note that a Procedure is different from a Process instance (see Process definition above).

4.2.4.3 Deployment

We begin this discussion with the SOSA/SSN definition of Deployment:

> “Describes the Deployment of one or more Systems for a particular purpose. Deployment may be done on a Platform.”

This is particularly important for systems such as unmanned systems (UxS) which might be deployed in one operating environment over one particular geography at a particular moment in time, and later deployed to another operating environment over a different geography at a different time. The DataStreams/Observations collected on these different Deployments may need to be tied to other mission data from a particular Deployment.

4.2.4.4 Sampling Feature

We begin this discussion with the SOSA/SSN definition of Sampling Feature. Sampling Feature is referred to as Sample in SOSA/SSN.

> "Sample - Feature which is intended to be representative of a FeatureOfInterest on which Observations may be made."

The OGC API - Connected Systems Standard v1.0 defines several sampling feature sub-types:

- Spatial Sampling Features
- Specimens (or Material Samples)
- Statistical Samples
- Feature Parts

Sampling Feature always refers to a larger Feature of Interest that they are a sample of. In the OGC API - Connected Systems Standard v1.0, any Feature can be a Feature of Interest, including Systems themselves. More detailed implementation guidance on Sampling Features will be provided in a future Part 4 extension to OGC API - Connected Systems Standard v1.0.

4.2.4.5 DataStream

We begin this discussion with the OGC API - Connected Systems Standard v1.0 definition of Data Stream, since there is no such SOSA/SSN definition:

> Data Stream is a particular type of ObservationCollection coming from a single System.

Note: An ObservationCollection in SOSA/SSN could include Observations from multiple different Systems. Among other things, a Data Stream provides the schema for the Result of Observations within the Data Stream.

4.2.4.6 Observation

We begin this discussion with the SOSA/SSN definition of Observation:

> “Act of carrying out an (Observation) Procedure to estimate or calculate a value of a property of a FeatureOfInterest. Links to a Sensor to describe what made the Observation and how; links to an ObservableProperty to describe what the result is an estimate of, and to a FeatureOfInterest to detail what that property was associated with.”

In the OGC API - Connected Systems Standard v1.0, Observations can have many different kinds of Result Types. And this is where we provide schemas for the Observation Result. This lets the API describe its Observations Types by providing a schema for each Data Stream, akin to how the OGC API - Features Standard provides schemas for each Feature Type.

4.2.4.7 ControlStream

We begin this discussion with the OGC API - Connected Systems Standard v1.0 definition of Control Stream, since there is no such SOSA/SSN definition:

> Control Stream defines the channels available for sending Commands to a given System.

Among other things, Control Streams provides schemas for the parameters for Commands within the Control Stream.

4.2.4.8 Command

We begin this discussion with the OGC API - Connected Systems Standard v1.0 definition of Command, since there is no such SOSA/SSN definition:

> Command carries the information required by a System to change the state of a Feature of Interest, which may be a System itself, a Subsystem of various Subtypes (e.g, Sensor, Process, Actuator, Platform, Sampler, etc.), or any other Feature.

In the OGC API - Connected Systems Standard v1.0, this is distinct from Actuation. The Command is not the actuation. And, a Command can control many different System Subtypes beyond Actuators. The Command is the information sent to control these various System Subtypes.

4.2.5. OGC Building Blocks

The OGC API - Connected Systems SWG is committed to reuse of OGC Building Blocks (https://blocks.ogc.org/) to the greatest extent possible. As this OGC Building Blocks process matures, the OGC API - Connected Systems Standard v1.0 may later reference Building Blocks external to the specification.

5. The OGC API - Connected Systems Standard v1.0 Encodings

The OGC API - Connected Systems Standard v1.0 supports a series of different encodings in order to enable particular kinds of functionality required by different communities. These encodings are based on the implementation models outlined in section 4.1.2 (above). These implementation models are based on the original XML encodings that have been at the core of the OGC Sensor Web Enablement architecture for the past two decades. The new OGC API - Connected Systems Standard v1.0 no longer requires the XML encodings of these implementation models (e.g., there is no conformance class for XML). Instead, the OGC API - Connected Systems Standard v1.0 relies on modern encodings such as JSON, Protobuf, Flatbuff, and other binary encodings could also be supported in extensions.

5.1. Ideas Driving Encoding Strategy

At the core of the OGC API - Connected Systems Standard v1.0's encoding strategy is the idea of reusing the concept of logical schemas from OGC API - Features specification to describe not only Feature properties but also Observation results. Under the legacy/heritage OGC SWE architecture, this was not possible due to the lack of alignment with the OGC API - Features Standard. This new alignment pays dividends in a number of ways. However, there is still the need for other encodings such as SensorML (and SWE Common Data Encoding Standard), to describe the sensing Systems themselves, and provide schemas for Observations within their Data Streams. However, now, there are more opportunities to align even between SensorML and the Feature model that will be explored in future versions of the OGC API - Connected Systems Standard v1.0 and the OGC API - Features Standard.

5.2. Different Kinds of Encodings

This section will walk you through the static Feature encodings and the encodings used for dynamic Data Streams (and Observations) and Control Streams (and Commands).

5.2.1 Static Feature Encodings

As an extension to the OGC API - Features Standard, the OGC API - Connected Systems Standard v1.0 is able to express a variety of things as static Features. These include the Systems themselves (and all of their GeoPose information), Procedures, Deployments, and Sampling Features.

5.2.1.1 SensorML

SensorML is used to provide detailed descriptions of Systems, Procedures and Deployments. SensorML 3.0 provides the ability to describe detailed characteristics, capabilities, and other metadata about these entities. Note: For Sampling Features, the specification just uses GeoJSON or JSON FG. This could evolve in future versions of SensorML.

5.2.1.2 GeoJSON/JSON FG

While SensorML is used to provide detailed descriptions, GeoJSON and JSON FG are used to provide summary descriptions when listing a large number of resources. JSON FG is required when the coordinate reference system is not CRS84 or CRS84h (e.g., WGS84 in Lon/Lat order).

5.2.1.3 Protobuf, FlatGeobuf

As mentioned above, there is ongoing work within the OGC API - Feature SWG to define Protobuf and FlatGeobuf encodings of Features and Geometries. Flatbuf could be used directly, but the OGC community has defined geospatial profiles of Flatbuf within FlatGeobuf. No equivalent standard geospatial profile exists for Protobuf at the time of writing.

5.2.2. Dynamic Data Stream Protocols and Encodings

Where the OGC API - Connected Systems Standard v1.0 extends beyond the OGC API - Features is with regard to dynamic Data Streams and Control Streams, to support real-time interactions within Systems of all kinds. This section addresses dynamic Data Streams.

5.2.2.1 Dynamic Data Stream Protocols

Protocols for dynamic Data Streams need to be lightweight and efficient. Oftentimes, specific technical communities have worked hard to define efficient protocols that can be used for streaming what can be voluminous streams of Observations. Protocols for implementing dynamic Data Streams within the OGC API - Connected Systems Standard v1.0 include:

5.2.2.1.1. WebSockets

Within the OGC API - Connected Systems Standard v1.0, WebSockets is used for Data Streams in the following ways:

- Retrieve real-time Observations from Data Streams (each WebSocket connection allows streaming data from a single Data Stream)

- Push real-time Observation into Data Streams (each WebSocket connection allows streaming data to a single Data Stream)

5.2.2.1.2. Pub/Sub protocols

Within the OGC API - Connected Systems Standard v1.0, Pub/Sub protocols for Data Streams are supported and advertised using AsyncAPI. They are used in the following ways:

- Subscribe to Observations from one or more Data Streams
- Publish Observations to Data Streams
- Subscribe to Data Stream resource events (creation/update/deletion events plus enable/disable events)

A future Part 3 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use such Pub/Sub mechanisms including MQTT, AMQP, DDS, Kafka, etc.

5.2.2.2 Dynamic Data Stream Encodings

Encodings for dynamic Data Streams need to be lightweight and efficient, and specialized for the specific Observation type. Oftentimes, specific technical communities have worked hard to define efficient JSON or XML encodings of their content, or efficient binary encodings for streaming what can be voluminous streams of data. Encodings for implementing dynamic Data Streams within the OGC API Connected Systems Standard v1.0 include:

5.2.2.2.1. JSON

Within the OGC API - Connected Systems Standard v1.0, various forms of JSON are used for Data Streams in the following ways:

- The Data Stream description itself is provided in JSON
- Logical schemas for Observation result and parameters are provided in SWE Common JSON
- Observation themselves can be encoded in JSON

The OGC API - Connected Systems Standard v1.0 does not model Data Streams/Observations as Features, and therefore does not use GeoJSON or JSON FG Features schema for Commands and Command Status - instead using JSON schemas.

5.2.2.2.2. Binary Encodings (Protobuf, Flatbuf, Apache Avro™ etc.)

For efficiency, the OGC API - Connected Systems Standard v1.0 also allows encoding Observations using binary formats such as Protobuf, Flatbuff or Apache Avro™ for example. When such binary encodings are used, an encoding specific schema is also provided (e.g. a proto file if Observations are encoded using Protobuf). Some binary encodings, such as H.264 can be implemented within Protobuf or Flatbuf streams or SWE Common binary encoded streams. A future Part 5 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use these binary formats.

FlatGeoBuf does have limitations as to its applicability to Observations (see 4.1.2.4 above), particularly with regard to video and other high bandwidth data types. This is why Flatbuf is referenced here rather than FlatGeoBuf, which is used to encode Features and Geometries (see below).

Beyond these generic binary encodings (e.g., Protobuf, Flatbuf, Apache Avro™, etc.) extensions can define additional binary formats for specific types of Observations like video, imagery, point clouds, etc. Or, more likely, an implementer can choose to reference another appropriate OGC interface that provides specialized format support such as OGC API - Coverage, OGC API - EDR, and OGC API - WAMI Best Practice.

Some of the benefits in using binary encodings like protobuf includes the use of a schema to define the structure of data. This schema is language-agnostic and can be used to generate code for various programming languages. This helps in maintaining a consistent data structure across different platforms. It is also self-describing and enforces strong typing which helps to catch data format errors at compile time. These binary encodings also has a smaller serialization overhead and because of its binary nature, it works well with compression algorithms. The binary nature of protobuf can enhance security by reducing the risk of injection attacks that are possible with text-based formats. However, it should be noted that protocol buffers is less human-readable than text-based formats and would hence be less suitable when human readability is essential.

5.2.3. Dynamic Control Stream Protocols and Encodings

Where the OGC API - Connected Systems Standard v1.0 extends beyond the OGC API - Features is with regard to dynamic Data Streams and Control Streams, to support real-time interactions within Systems of all kinds. This section addresses dynamic Control Streams.

5.2.3.1 Dynamic Control Stream Protocols

Protocols for dynamic Control Streams need to be lightweight and efficient. Often times, specific technical communities have worked hard to define efficient protocols that can be used for controlling Systems with voluminous streams of Commands. Protocols for implementing dynamic Control Streams within the OGC API Connected Systems Standard v1.0 include:

5.2.3.1.1. WebSockets

Within the OGC API - Connected Systems Standard v1.0, WebSockets is for Control Streams in the following ways:

- Push real-time Commands into Control Streams, and receive ACK (each WebSocket connection allows streaming data to a single Control Stream)
- Retrieve real-time Commands from Control Streams (each WebSocket connection allows streaming data from a single Control Stream)

5.2.3.1.2. Pub/Sub protocols

Within the OGC API - Connected Systems Standard v1.0, Pub/Sub protocols for Control Streams are supported and advertised using AsyncAPI. They are used in the following ways:

- Publish Commands to Control Streams
- Subscribe to Command status messages (i.e. initial ACK, status report for long running commands, etc.)
- Subscribe to Commands received from one or more Control Streams
- Subscribe to Control Stream resource events (creation/update/deletion events plus enable/disable events)

A future Part 3 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use such Pub/Sub mechanisms including MQTT, AMQP, DDS, Kafka, etc.

5.2.3.2. Dynamic Control Stream Encodings

Encodings for dynamic data streams need to be lightweight and efficient, and specialized for the specific Observation type. Often times, specific technical communities have worked hard to define efficient JSON or XML encodings of their content, or efficient binary encodings for streaming what can be voluminous streams of data. Encodings for implementing dynamic Control Streams within the OGC API Connected Systems Standard v1.0 include:

5.2.3.2.1 JSON (GeoJSON/JSON FG)

Within the OGC API - Connected Systems Standard v1.0, various forms of JSON are used for Control Streams in the following ways:

- The Control Stream description itself is provided in JSON
- Logical schemas for Command parameters and results are provided in SWE Common JSON
- Command themselves can be encoded in JSON

The OGC API - Connected Systems Standard v1.0 does not model Control Streams/Commands as Features, and therefore does not use GeoJSON or JSON FG Features schema for Commands and Command Status - instead using JSON schemas.

5.2.3.2.2 Binary Encodings (Protobuf, Flatbuf, Apache Avro™ etc.)

For efficiency, the OGC API - Connected Systems Standards v1.0 also allows encoding Commands, Command Status and Command Results using binary formats such as Protobuf, Flatbuff or Apache Avro™ for example. When such binary encodings are used, an encoding specific schema is also provided (e.g. a proto file if Protobuf is used). A future Part 5 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use these binary formats.

Note: FlatGeoBuf does have limitations as to its applicability to real-time Commands (see 4.1.2.4 above). This is why Flatbuf is referenced here rather than FlatGeoBuf, which is used to encode Features and Geometries (see below).

6. OGC API - Connected Systems Standard v1.0 in the Landscape of Standards

It is always the goal to have an orderly set of interoperability standards with a clear set of relationships between each other, and no ambiguity or minimal overlap in their functionality. However, not only are there similar looking standards built for different purposes, but they often interact with each other in useful and surprising ways.

This discussion of how the OGC API - Connected Systems Standard v1.0 sits within the larger landscape of standards will be provided in 3 parts. First, we will discuss how the OGC API - Connected Systems Standard v1.0 relates to other OGC Standards. Second, we will discuss how it relates to other Web standards. And Third, we will discuss other standards that are related, but which the OGC API - Connected Systems Standard does not normatively reference or link to.

6.1. OGC Universe of Standards

Within the OGC's universe of standards, there are complementarities, touch points, and isomorphic functions. With the OGC API - Connected Systems Standard v1.0, things are no different. This discussion will address those OGC specifications that are normatively referenced within the OGC API - Connected Systems Standard v1.0, and those that the OGC API - Connected Systems Standard v1.0 links with, since all OGC API based standards are built on the same patterns, we can combine functionality from different OGC APIs on the same endpoint.

6.1.1 Normatively Referenced OGC Standards

The OGC specifications that are normatively referenced within the OGC API - Connected Systems Standard v1.0 are:

6.1.1.1 OGC API - Features (Part 1, Part 3, Part 4)

As mentioned above, if a user (e.g., human or process) seeks a static Feature representation of Systems published by a given instance of the OGC API - Connected Systems Standard v1.0, they will access this static Feature representation from the OGC API - Features part (e.g., Part 1) of the specification. It is also possible that this static Feature representation might be served by a linked remote OGC API - Feature instance. For more on this later point regarding linked resources, see section 6.1.7 below. Note: Part 2 related to coordinate reference Systems can also be used within the OGC API - Connected Systems Standard v1.0. Also Note: The draft

Part 5 addresses issues of how Pub/Sub mechanisms will be addressed in OGC API - Features. See discussion in section 6.1.1.2 below.

6.1.1.2 OGC API - Pub/Sub

Given that several OGC API SWGs, including the OGC API - Connected Systems SWG, contemporaneously expressed interest in adopting a common architecture for asynchronous communication, and support the approach proposed by the OGC API - EDR SWG - an approach based on the AsyncAPI (which is the asynchronous counterpart to OpenAPI). This approach supports the asynchronous workflows of the previous OGC SWE interface while better conforming with the OpenAPI strategic guidance provided by the OGC API - Connected Systems Standard v1.0. This approach is also being adopted by the OGC API - Features SWG in their draft Part 5.

6.1.1.3. SOSA/SSN/OMS

As mentioned above, SOSA/SSN/OMS is one of the core conceptual level information models underpinning the OGC API - Connected Systems Standard v1.0. Its predecessor, the OGC SWE specification, predated the SOSA/SSN/OMS standard, but provided support for all of the concepts that have been formalized within the SOSA/SSN/OMS specification. See Section 3: Terms and Definitions for more detail.

6.1.1.4. SensorML

As mentioned above, SensorML v3.0 is one of the core implementation level information models underpinning the OGC API - Connected Systems Standard v1.0, and its predecessor, the OGC SWE Standards. See Section 3: Terms and Definitions for more detail.

6.1.1.5. SWE Common Data Model Encoding Standard

As mentioned above, SWE Common Data Model Encoding Standard v3.0 is one of the core implementation level information models underpinning the OGC API - Connected Systems Standard v1.0, and its predecessor, the OGC SWE Standards. See Section 3: Terms and Definitions for more detail.

6.1.1.6. GeoPose

As mentioned above, GeoPose is one of the core conceptual level information models underpinning the OGC API - Connected Systems Standard v1.0. Its predecessor, the OGC SWE Standards, predated the GeoPose standard, but provided support for all of the concepts that have been formalized within the GeoPose specification. See Section 3: Terms and Definitions for more detail.

6.1.2 Linking to external Observation result

As mentioned above in section 6.1.1., if the user seeks to have Observations provisioned in the form of static Features, by an external OGC API - Features instance, they can request data from

Part 1 of the OGC API - Connected Systems Standard v2.0, which conforms to the OGC API - Features specification, or from a remote OGC API - Features instance.

Beyond this, the other OGC specifications that the OGC API - Connected Systems Standard v1.0 links with (organized according to their OGC API - Connected Systems function) include:

6.1.2.1. Link to OGC API - Maps

If the user seeks a map in response to their request for Observations, it can be provided through a link to an OGC API - Maps interface.

6.1.2.2. Link to OGC API - Coverages

If the user seeks to request Observations in the form of a gridded coverage, or to further slide and dice raster observations (aka gridded coverages) it can be provided through a link to an OGC API - Coverage interface.

6.1.2.3. Link to OGC API - EDR

If the user seeks to discover or query data resources from an OGC API - Environmental Data Retrieval (EDR) instance, they can request metadata about the resources provided by the server, or execute query operations to retrieve EDR resources from the underlying data store based upon simple selection criteria, defined by this standard and selected by the client. This can include subsetting certain Connected Systems resources available through an OGC API - EDR instance.

6.1.2.4. Link to OGC SensorThingsAPI

If the user seeks to request Observations from an OGC SensorThings API that is linked to an instance of the OGC API - Connected Systems Standard v1.0, they can do so.

6.1.2.5. Link to OGC API - 3D Volumes/3D Tiles

If the user seeks to request 3D Features of Interest response to their request for Observations, it can be provided through a link to an OGC API - 3D Volumes or 3DTiles interface.

6.1.2.6. Link to API OGC - Records

If the user seeks to request metadata records regarding a System, or the Observations from a particular System, it can be provided through a link to an OGC API - Records interface.

6.1.2.7. Link to OGC API - Moving Features

If the user seeks to request OGC API - Moving Features response to their request for Observations, they can do so.

6.1.2.8. Link to OGC WAMI Best Practice

If the user seeks to request OGC WAMI Best Practice response to their request for Observations, they can do so.

***

Future support for linking to yet to be approved OGC specifications such as GeoDCAT (https://www.ogc.org/press-release/ogc-forms-new-geodcat-standards-working-group/), based on the W3C's Data Catalog Vocabulary (https://www.w3.org/TR/vocab-dcat-3/) can also be added.

6.2. Other Web Standards

The OGC has long held Class A liaison relationships with other international standards organizations (ISO) that promulgate Web standards, and other domain standards. Some of these are normatively referenced through various OGC API Standards, including

6.2.1. OpenAPI (https://www.openapis.org/)

As mentioned above, the scope for the OGC API - Connected Systems Standard v1.0 was very much defined by the OGC's strategic guidance to migrate all legacy/heritage specifications to OpenAPI/RESTful patterns.

6.2.2. AsyncAPI (https://www.asyncapi.com/)

As mentioned above, AsyncAPI (which is the asynchronous counterpart to OpenAPI) provides the asynchronous workflows of the previous OGC SWE interface while better conforming with the OGC Architecture Board's strategic guidance. AsynchAPI is used in OGC API - Connected Systems Standard v1.0 to define asynchronous and Pub/Sub interfaces.

6.2.3. JSON (https://www.json.org/)

JSON, also known as ECMA-404 The JSON data interchange syntax (2nd edition, December 2017) was published by Ecma International (https://www.ecma-international.org/publications-and-standards/standards/ecma-404/) which, since 1961 facilitates the timely creation of a wide range of global Information and Communications Technology (ICT) and Consumer Electronics (CE) standards. JSON is flexible and powerful format, which can be profiled in innovative ways. It underpins GeoJSON and JSON-FG, which, despite their common reliance on JSON, diverge on important issues such as spatial reference system/coordinate system support. The OGC API - Connected Systems Standard v1.0 utilizes JSON as its main resource format.

6.2.4. XML (https://www.w3.org/XML/)

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879), managed by the World Wide Web Consortium's XML Activity (www.w3.org). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. The use of XML is deprecated in the OGC API - Connected Systems Standard v1.0.

6.2.5. Protobuf (https://protobuf.dev/)

Protocol Buffers are language-neutral, platform-neutral extensible mechanisms for serializing structured data. One uses Protobufs if they want to be more efficient and the message is not that big (1 MB or less). The binary nature of protobuf can enhance security by reducing the risk of injection attacks that are possible with text-based formats. Protobuf is used in OGC API - Connected Systems Standard v1.0 as an efficient resource format. A future Part 5 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use these binary formats.

6.2.6. Flatbuf / FlatGeoBuf (https://flatbuffers.dev, http://flatgeobuf.org/, https://www.ogc.org/tag/flatgeobuf/)

Use FlatBuffers if we want to be more efficient with larger messages. FlatBuffers is the better choice if you're looking to create read-only query messages - this feature also saves on time and memory. A future Part 5 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use these binary formats.

Note: FlatGeoBuf does have limitations as to its applicability to real-time Commands (see 4.1.2.4 above). This is why Flatbuf is referenced here rather than FlatGeoBuf, which is used to encode Features and Geometries (see below).

6.2.7. Apache Avro™ (https://avro.apache.org/)

Apache Avro™ is the leading serialization format for record data, and first choice for streaming data pipelines. It offers excellent schema evolution, and has implementations for the JVM (Java, Kotlin, Scala, …), Python, C/C++/C#, PHP, Ruby, Rust, JavaScript, and even Perl. A future Part 5 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use these binary formats.

6.2.8. Message Queuing Telemetry Transport (MQTT)

MQTT (https://mqtt.org/) is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish and subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. MQTT today is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas, etc. The OGC API - Connected System Standard v1.0 utilizes MQTT for Data Streams and Control Streams. MQTT is hub-and-spoke and is optimized for centralized data collection and analysis – connecting sensors and mobile devices to applications or a message broker. A future Part 3 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use such Pub/Sub mechanisms.

6.2.9. Advanced Message Queuing Protocol (AMQP)

AMQP (https://www.amqp.org) is an open-standard protocol for message-oriented middleware that enables communication between different software components in a distributed system. AMQP is designed to facilitate the efficient and reliable exchange of messages between applications or systems, making it a valuable tool for building scalable and robust messaging solutions. It is commonly used in scenarios such as messaging systems, queueing systems, and inter-process communication in various software applications.

- On top of the publish and subscribe messaging patterns, it allows for point-to-point and request-reply scenarios. These patterns allow for flexibility in designing communication between components.
- Various properties, including headers, body and routing information. These properties are used for message filtering and routing.
- Security features such as authentication and authorization. It also supports transport layer security (TLS/SSL) for secure communication.
- Flow control to prevent message congestion and ensure that consumers receive messages at a manageable rate.
- Presence of routing rules allows determination of how messages are delivered to queues and ultimately, to consumers.
- Error codes and mechanisms during message exchange helps in building robust and reliable messaging systems.

A future Part 3 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use such Pub/Sub mechanisms.

6.2.10. Data Distribution System (DDS)

The Object Management Group (OMG) Data Distribution Service for Real-Time Systems (DDS) standard (https://www.omg.org/omg-dds-portal/) was designed specifically to address machine-to-machine (M2M) communication, directly connecting sensors, devices and applications to each other without any dependence on centralized IT infrastructure. While DDS is not called out explicitly in the OGC API - Connected Systems Standard, it can be accommodated as an extension. A future Part 3 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use such Pub/Sub mechanisms.

6.2.11. Kafka

Kafka (https://kafka.apache.org/intro) is a distributed system consisting of servers and clients that communicate via a high-performance TCP network protocol. It can be deployed on bare-metal hardware, virtual machines, and containers in on-premise as well as cloud environments. A future Part 3 extension of OGC API - Connected Systems Standard v1.0 will provide implementation details on how to use such Pub/Sub mechanisms.

6.2.12. Web of Things (WoT)

The Web of Things (WoT) (https://www.w3.org/WoT/) seeks to counter the fragmentation of the IoT by using and extending existing, standardized Web technologies. By providing standardized metadata and other re-usable technological building blocks, W3C WoT enables easy integration across IoT platforms and application domains. OGC data models (including OMS, SensorML and SWE Common) provide a superset of WoT capabilities. It should always be possible to translate a WoT Thing Description (TD) to a SensorML description for example (the reverse is true too but potentially with some loss of information). An implementation of Connected Systems can be used to integrate with WoT.

6.3. Related Standards

There are many related standards that are not normatively referenced in the OGC API - Connected Systems Standard v1.0.

6.3.1 Related Geospatial Format Standards

There are many related format standards that are not normatively referenced in the OGC API - Connected Systems specification

6.3.1.1 H. 264/MISB/STANAG 4609

(https://gwg.nga.mil/gwg/focus-groups/Motion_Imagery_Standards_Board_(MISB).html)

H. 264, also called Advanced Video Coding (AVC), is the most common video compression standard in use today. When used for overhead imagery from drones and satellites, telemetry data can be encoded in Motion Imagery Standards Board (MISB) metadata within H.264. STANAG 4609 describes an exchange format for motion imagery. It is the official format for motion imagery (video data, image sequences, FMV - full motion videos) exchange within the NATO nations. Motion imagery is defined by MISB to be video of at least 1 Hz image frequency together with metadata. STANAG 4609 describes the encoding of the video and the metadata (geographical data) for different usages. This includes the supported video codecs, bit rates, frame rates, container formats, metadata content, metadata encoding and hardware to distribute the motion imagery.

6.3.1.2 STAC Item (https://stacspec.org)

SpatioTemporal Asset Catalog (STAC) specification provides a common structure for describing and cataloging spatiotemporal assets. A STAC Item is the core atomic unit, representing a single spatiotemporal asset as a GeoJSON feature plus datetime and links.

6.3.1.3 COG (https://www.cogeo.org/)

A Cloud Optimized GeoTIFF (COG) is a regular GeoTIFF file, aimed at being hosted on a HTTP file server, with an internal organization that enables more efficient workflows on the cloud. It does this by leveraging the ability of clients issuing HTTP GET range requests to ask for just the parts of a file they need.

6.3.1.4 LAS (https://www.asprs.org/wp-content/uploads/2019/07/LAS_1_4_r15.pdf)

The LAS file format is a public file format for the interchange of 3-dimensional point cloud data data between data users. Although developed primarily for exchange of LiDAR point cloud data, this format supports the exchange of any 3-dimensional x,y,z tuplet. LAS is a Standard of the American Society for Photogrammetry & Remote Sensing.

6.3.1.5 Gridded Coverage/Imagery Formats

There are countless other gridded coverage and imagery formats that are commonly generated by Sensors of all kinds. The OGC API - Connected Systems Standard provides support for any and all of these. This includes GRIB, NetCDF, HDF, HDF-EOS, JPEG, JPEG2000, GRASS, NITF, and Compensated Phase History Data (CPHD). (File format versions are intentionally excluded).

6.3.2. Related Libraries and Interface Standards

There are many related interface standards that are not normatively referenced in the OGC API - Connected Systems Standard v1.0 and cannot be linked, but which have shared and overlapping purpose.

6.3.2.1 ArduPilot (https://ardupilot.org/)

ArduPilot is an open source, unmanned vehicle autopilot software suite capable of controlling autonomous multirotor drones, fixed-wing and VTOL aircraft, helicopters, ground rovers, boats, submarines, antenna trackers. ArduPilot was originally developed by hobbyists to control model aircraft and rovers and has evolved into a full-featured and reliable autopilot used by industry, research organizations, and amateurs. (https://en.m.wikipedia.org/wiki/ArduPilot)

As the dominant standard for UxS remote piloting and autopiloting, ArduPilot serves as one of the primary bridges from which any OGC API - Connected Systems based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control UxS. Preliminary mappings demonstrate compatibility between these two standards.

6.3.2.2 Distributed Common Ground System (DCGS)

The Distributed Common Ground System (DCGS) is a system of the U.S. Department of Defense which produces military intelligence for multiple military branches, combatant commands, and combat support agencies. Each has their own DCGS segments capable of sharing information between each via DCGS core services.

6.3.2.3 Defense Intelligence Information Enterprise (DI2E)

DI2E is the component of the U.S. Defense Intelligence Enterprise that a) transforms information collected for intelligence needs into forms suitable for further analysis and action; b) provides the ability to integrate, evaluate, interpret and predict the current and future operations/physical environment; and c) provides the ability to present, distribute or make available intelligence, information and environmental content and products that enable better situational awareness to military and national decision making.

6.3.2.4 DJI SDK (https://www.dji.com/)

DJI, a global technology company that specializes in manufacturing drones and aerial photography systems, is one of the leading manufacturers of drones, with a significant market share in the industry.

DJI SDK (Software Development Kit) is a set of software development tools and resources. The DJI SDK allows developers to create custom applications that can interact with DJI's drones and other products.

The DJI SDK provides access to a range of features and functions, including flight control, camera control, and data transmission. Developers can use the SDK to create custom applications for a variety of use cases, such as aerial photography, surveying, mapping, and inspection.

As the proprietary mechanism for interfacing with all DJI drones, this serves as one of the primary bridges from which any OGC API - Connected Systems Standard v1.0 based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control DJI Platforms and Sensors. Preliminary mappings demonstrate compatibility between these two specifications.

6.3.2.5 SAPIENT (https://www.gov.uk/guidance/sapient-autonomous-sensor-system)

Sensing for Asset Protection with Integrated Electronic Networked Technology (SAPIENT) uses autonomy to reduce the workload of people operating multi-sensor systems, in security and defence scenarios. It is the concept of a network of advanced sensors with artificial intelligence (AI) at the edge, combined with intelligent fusion and sensor management. The benefits of SAPIENT include:

- significantly lower cognitive burden on operators
- lower communications bandwidth
- operational flexibility
- dual defence and security use
- lower acquisition cost

SAPIENT has been adopted by MOD as the standard for counter-UAS(uncrewed air system) technology. It is also being evaluated as a potential NATO standard for counter-drone systems. Preliminary mappings demonstrate compatibility between SAPIENT and OGC API - Connected Systems Standard v1.0.

6.3.2.6 Integrated Sensor Architecture (ISA - https://apps.dtic.mil/sti/pdfs/AD1079785.pdf)

ISA is a U.S. Army Service-Oriented Architecture (SOA) developed by the Night Vision Electronic Sensors Directorate (NVESD) of what now is the US Army DevCom C5ISR Center. ISA identifies common standards and protocols, which support a net-centric system-of-systems integration. Utilizing a common language, these systems are able to connect, publish their needs and capabilities, and interact dynamically. ISA provides an extensible data model with defined capabilities, and provides a scalable approach across multi-echelon deployments, which when coupled with dynamic discovery capabilities, cybersecurity, and sensor management, provides a system which can adjust and adapt to dynamic environment. ISA capabilities enable Soldiers to exchange information between their own sensors and those on other Platforms in a fully dynamic and shared environment. ISA enables Army sensors and systems to readily integrate into an existing network and dynamically share information and capabilities to improve situational awareness in a battlefield environment.

As the dominant standard for discovering, accessing, visualizing and tasking sensors on US Army Platforms, ISA serves as one of the primary bridges from which any OGC API - Connected Systems Standard v1.0 based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control ISA Sensors. Preliminary mappings demonstrate compatibility between these two standards.

6.3.2.7 Joint Interface Control Document (JICD) 4.2.1

The JICD for common services lets systems become interoperable with Network-Centric Collaborative Targeting (NCCT) and Theater Net-Centric Geolocation (TNG) sensor fusion networks.

As a niche Department of Defense (DOD) standard for discovering, accessing, visualizing and tasking Electronic Warfare systems, JICD 4.2.1 serves as one of the primary bridges from which any OGC API - Connected Systems Standard v1.0 based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control JICD 4.2.1 Systems. Preliminary mappings demonstrate compatibility between these two standards.

6.3.2.8 Micro Air Vehicle Link (MavLink - https://mavlink.io)

MAVLink is a protocol for communicating with small unmanned vehicle. It is designed as a header-only message marshaling library. MAVLink was first released early 2009 by Lorenz Meier under the LGPL license.
(https://en.m.wikipedia.org/wiki/MAVLink)

As the dominant standard for communicating with UxS, MAVLink serves as one of the primary bridges from which any OGC API - Connected Systems Standard v1.0 based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control UxS. Preliminary mappings demonstrate compatibility between these two standards.

6.3.2.9 Robot Operating System (ROS - https://ros.org/)

ROS is an open-source robotics middleware suite. Although ROS is not an operating system (OS) but a set of software frameworks for robot software development, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post, and multiplex sensor data, control, state, planning, actuator, and other messages. Despite the importance of reactivity and low latency in robot control, ROS is not a real-time operating system (RTOS). However, it is possible to integrate ROS with real-time computing code.[3] The lack of support for real-time systems has been addressed in the creation of ROS 2,[4][5][6] a major revision of the ROS API which will take advantage of modern libraries and technologies for core ROS functions and add support for real-time code and embedded system hardware. (https://en.m.wikipedia.org/wiki/Robot_Operating_System)

As the dominant standard for controlling robots, ROS serves as one of the primary bridges from which any OGC API - Connected Systems Standard v1.0 based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control ROS based robotic Platforms. Preliminary mappings demonstrate compatibility between these two standards.

6.3.2.10 Sensor Open Systems Architecture (SOSA - http://prod.opengroup.org/sosa)

SOSA, developed by the OpenGroup, establishes guidelines for Command, Control, Communications, Computers, Cyber, Intelligence, Surveillance and Reconnaissance (C5ISR) systems. The objective is to allow flexibility in the selection and acquisition of sensors and Subsystems that provide sensor data collection, processing, exploitation, communication, and related functions over the full life cycle of the C5ISR system.

As a dominant hardware standard for connecting sensors into larger C5ISR systems, SOSA serves as one of the primary bridges from which any OGC API - Connected Systems Standard v1.0 based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control such Sensor Systems. Preliminary mappings demonstrate compatibility between these two standards.

6.3.2.11 SISO High Level Architecture (HLA) and Distributed Interactive Simulation (DIS) (https://www.sisostds.org/StandardsActivities/DevelopmentGroups/HLAPDG-High-Level Architecture.aspx, and https://www.sisostds.org/StandardsActivities/SupportGroups/DISRPRFOMPSG.aspx)

In 1995, the Defense Modeling and Simulation Office (DMSO) formulated a vision for modeling and simulation and established a modeling and simulation masterplan, which included the High Level Architecture (HLA). The purpose of HLA is to provide one unified standard that would meet the simulation interoperability requirements of all US DoD components, and to support legacy modeling and simulation interoperability protocols including the Distributed Interactive Simulation (DIS) protocol. To facilitate usage outside of the defense community, HLA was then transitioned into an IEEE standard, maintained by Simulation Interoperability Standards Organization (SISO). SISO-PN-016-2016 established the High Level Architecture (HLA) Product Development Group which developed and maintains High-Level Architecture Version 3.0. The PDG operates simultaneously as the HLA Working Group under the IEEE Computer Society Standards Activities Board Simulation Interoperability (C/SI) SISO SAC Standards Committee.

To facilitate the migration for DIS users, a Federation Object Model corresponding to the fixed object model of DIS was also developed as the Real-time Platform Reference FOM (RPR FOM). The Distributed Interactive Simulation / Real-time Platform Reference Federation Object Model (DIS / RPR FOM) Product Support Group (PSG) is a permanent support group chartered by the Simulation Interoperability Standards Organization (SISO) Standards Activity Committee to support multiple DIS-related products including:

- IEEE Std 1278.1™-2012, IEEE Standard for Distributed Interactive Simulation - Application Protocols (a revision of IEEE Std 1278.1™-1995 and IEEE Std 1278.1a™-1998)
- IEEE Std 1278.2™-2015, IEEE Standard for Distributed Interactive Simulation (DIS) - Communication Services and Profiles (a revision of IEEE Std 1278.2™-1995)
- IEEE Std 1278.4™-1997, IEEE Recommended Practice for Distributed Interactive Simulation - Verification, Validation, and Accreditation
- SISO-STD-001-2015, Standard for Guidance, Rationale, and Interoperability Modalities (GRIM) for the Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0
- SISO-STD-001.1-2015, Standard for Real-time Platform Reference Federation Object Model (RPR FOM), Version 2.0

As a dominant standard for connecting interactive simulations to larger systems, SISO HLA/DIS offers OGC API - Connected Systems Standard v1.0 based systems an opportunity to integrate simulated data feeds into larger systems for many purposes, including mission planning and rehearsal, as well as the inclusion of simulations of phenomena that may not be observable in real time, in order to calibrate real time operations. As such, SISA HLA/DIS serve as one of the primary bridges from which any OGC API - Connected Systems based System will receive Data Streams of Observations, and over which it would send Control Streams (e.g., feasibility and tasking commands) to control such simulations. Preliminary assessments demonstrate compatibility between these two standards.

6.3.2.12 Spatio-Temporal Asset Catalog (STAC - https://stacspec.org)

The STAC specification is a common language to describe geospatial information, so it can more easily be worked with, indexed, and discovered. Though it began independently, the current version of STAC is based on the OGC API - Features specification, where a “STAC item” is a Feature.

As a dominant standard for managing remote sensing imagery archives, emerging STAC based tasking/ordering strategies (currently called Spatio-Tempora Asset Tasking - STAT) offer OGC API - Connected Systems specification based feasibility and tasking commands a potential bridge to traverse in order to order data collection from remote sensing satellite constellations. STAC is already aligned with the OGC API - Features specification, and the STAC community has expressed its desire for its future tasking/ordering interface to continue to OGC API standards. The OGC API - Connected Systems editors are committed to remaining engaged in the STAT process.

6.3.2.13 Universal C2 Language (UC2 - https://www.sei.cmu.edu/publications/annual-reviews/2021-year-in-review/year_in_review_article.cfm?customel_datapageid_315013=335863)

The UC2 program comprises a set of technical working groups led by a coalition of six federally funded research and development centers (FFRDCs) with representatives from the military. Together, Fully Networked Command, Control, and Communications (FNC3) and the Aerospace Corporation, the Institute for Defense Analyses Systems and Analyses Center, the MIT Lincoln Laboratory, the MITRE National Security Engineering Center, the RAND National Defense Research Institute, and the SEI are developing a universal C2 language and standard.

As an emerging and evolving specification, UC2 is similar to many other DoD specifications for command and control objects, and should easily be accommodated within the OGC API - Connected Systems Standard v1.0 framework. Unless there is a serious regression in C2 language from existing C2 capabilities, compatibility between these two standards should be straightforward.

6.3.2.14 Universal Command and Control Interface (UCI)

UCI is a standard managed, systematized and evolved by the US Air Force’s Open Architecture Management Standards (OAMS) and the Open Mission Systems (OMS) standard. The OAMS enable current, legacy, and new programs to realize the benefits of Open Architecture.

The USAF’s UCI standard is one of several “universal” C2 standards from the US DoD. As with UCI, unless there is a serious regression in C2 language from existing C2 capabilities, compatibility between UCI and the OGC API - Connected Systems Standard v1.0 should be straightforward.

6.3.2.15 Web Graphics Library (WebGL)

WebGL is a web technology standard that defines a JavaScript API for rendering 2D and 3D graphics within web browsers. WebGL would be an integral part to support the dynamic data streaming of the OGC API on applications, and where interative and visually rich graphics and animations with the Geospatial web applications are use cases. References to WebGL standards could be found in Khronos Group (www.khronos.org/api/webgl) or any web browser platforms that use WebGL.

6.3.2.16 Universal Scene Description (USD)

USD provides for interchange of 3D elemental assets or animations, and allows assembly and organisation of elemental assets into 3D scenes in a robust manner. It is used in many 3D content creation applications, to compose scenes across different file formats and enable live, collaborative scene construction (https://openusd.org/release/intro.html).

6.3.2.17 WebGPU

WebGPU exposes an API for performing operations, such as rendering and computation, on a Graphics Processing Unit. (https://www.w3.org/TR/webgpu/)

7. Use Cases

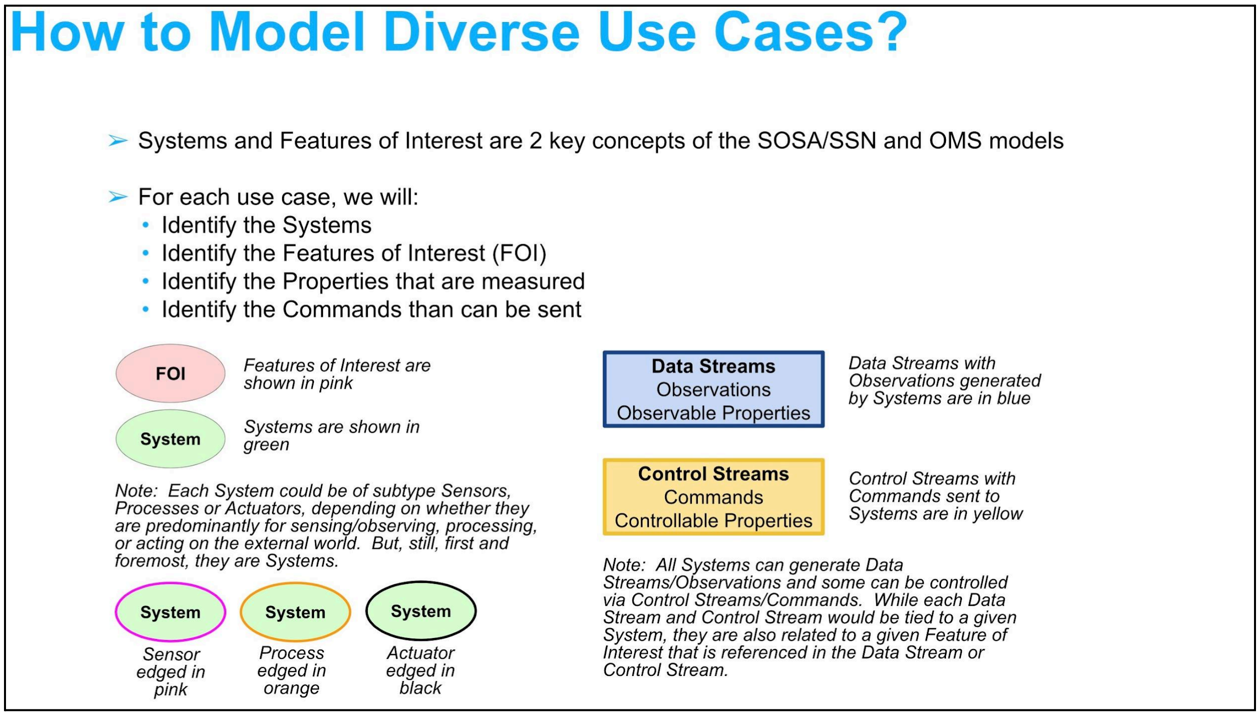
Interoperability specifications such as the OGC API - Connected Systems Standard v1.0 can only truly be understood when seen through the lens of concrete, real world examples. This section provides a series of technical use cases and a series of domain use cases. Together, reviewers should be able to better understand how Systems, Platforms, Sensors, Processes, Actuators, Features of Interest, Data Streams and their Observations, and Control Streams and their Commands work together within the OGC API - Connected Systems Standard v1.0.

7.1. Technical use cases

This section provides concrete technical use cases of how Systems, Platforms, Sensors, Processes, Actuators, Features of Interest, Data Streams and their Observations, and Control Streams and their Commands work together when integrating different kinds of systems via the OGC API - Connected Systems Standard v1.0. These include, but are not limited to:

1) IoT Thing
2) Weather Station
3) Pan Tilt Zoom (PTZ) Camera

4) Aircraft Telemetry
5) Ground Vehicle
6) Surface Vessel
7) Unmanned Aerial Vehicle (Aerial UxS)
8) Unmanned Ground Vehicle (Ground UxS)
9) Unmanned Surface Vehicle (Surface Marine UxS)
10) Unmanned Underwater Vehicle (Underwater Marine UxS)
11) Spaceborne Systems
12) Cell Towers
13) GMTI SAR
14) Air Traffic Radar
15) Doppler Radar
16) Counter UAS Radar
17) Weather Forecast Model
18) Flight Optimization
19) Tipping and Cueing (Laser Range Finder to PTZ)
20) Alerts/Notification (Temperature Threshold)
21) Cyber Sensor
22) Human as Sensor
23) Human as Platform
24) Human Receiving Command
25) Dynamic Data Feed



How to Model Diverse Use Cases?

➢ Systems and Features of Interest are 2 key concepts of the SOSA/SSN and OMS models

➢ For each use case, we will:
- Identify the Systems
- Identify the Features of Interest (FOI)
- Identify the Properties that are measured
- Identify the Commands than can be sent

FOI *Features of Interest are shown in pink*

System *Systems are shown in green*

Note: Each System could be of subtype Sensors, Processes or Actuators, depending on whether they are predominantly for sensing/observing, processing, or acting on the external world. But, still, first and foremost, they are Systems.

System *Sensor edged in pink*

System *Process edged in orange*

System *Actuator edged in black*

Data Streams
Observations
Observable Properties

Data Streams with Observations generated by Systems are in blue

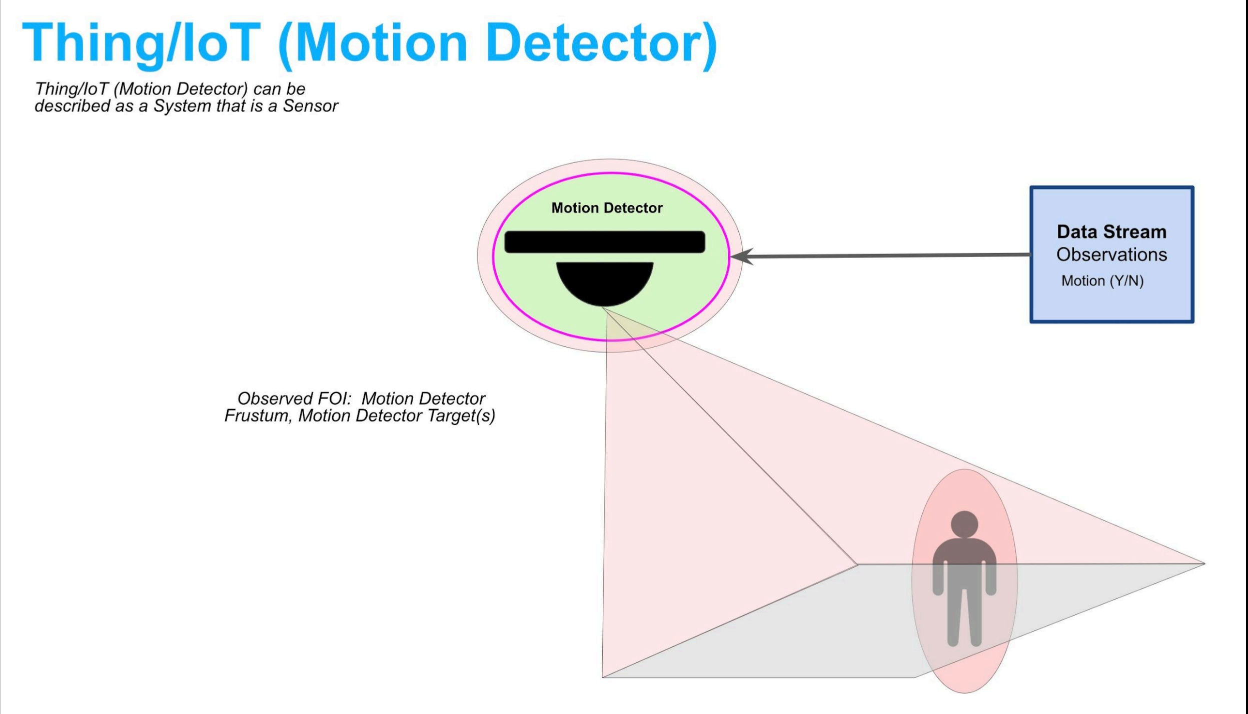
Control Streams
Commands
Controllable Properties

Control Streams with Commands sent to Systems are in yellow

Note: All Systems can generate Data Streams/Observations and some can be controlled via Control Streams/Commands. While each Data Stream and Control Stream would be tied to a given System, they are also related to a given Feature of Interest that is referenced in the Data Stream or Control Stream.

7.1.1. Thing/IoT (Motion Detector)

When a sensing System has a single purpose, it is often termed a “Thing”, as part of the Internet of Things. This Thing (IoT) example is of a Motion Detector, which is a Sensor. Other such Things could be Actuators, such as electronic door locks. As everything becomes connected to the Internet, creating the “Internet of Everything”, Things are becoming more and more complex. Still this example seeks to showcase a simple System of SubType Sensor. The diagram and discussion below help convey how IoT Things can be treated in the OGC API - Connected Systems Standard v1.0.


Thing/IoT (Motion Detector)
Thing/IoT (Motion Detector) can be described as a System that is a Sensor
Motion Detector
Data Stream
Observations
Motion (Y/N)
Observed FOI: Motion Detector Frustum, Motion Detector Target(s)

System: The top level System is the Sensor in this case
Platform: None
Sensors: Motion Detector
Processes: None
Actuators: None
Features of Interest: Motion Detector Frustum, Object(s)
Data Streams/Observations: Motion Detection (Y/N)
Control Streams/Commands: None

Systems:

Name	Type	Description (+ link to datasheet)
Thing (Motion Detector)	Sensor	

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Motion Detector Frustum	Motion Detector	The volume covered by the detector
Object(s)	Motion Detector	The objects whose motion is being detected.

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

System	**Data Stream**	**Comments**
Motion Detector	Motion (Y/N)	

Control Streams/Commands:

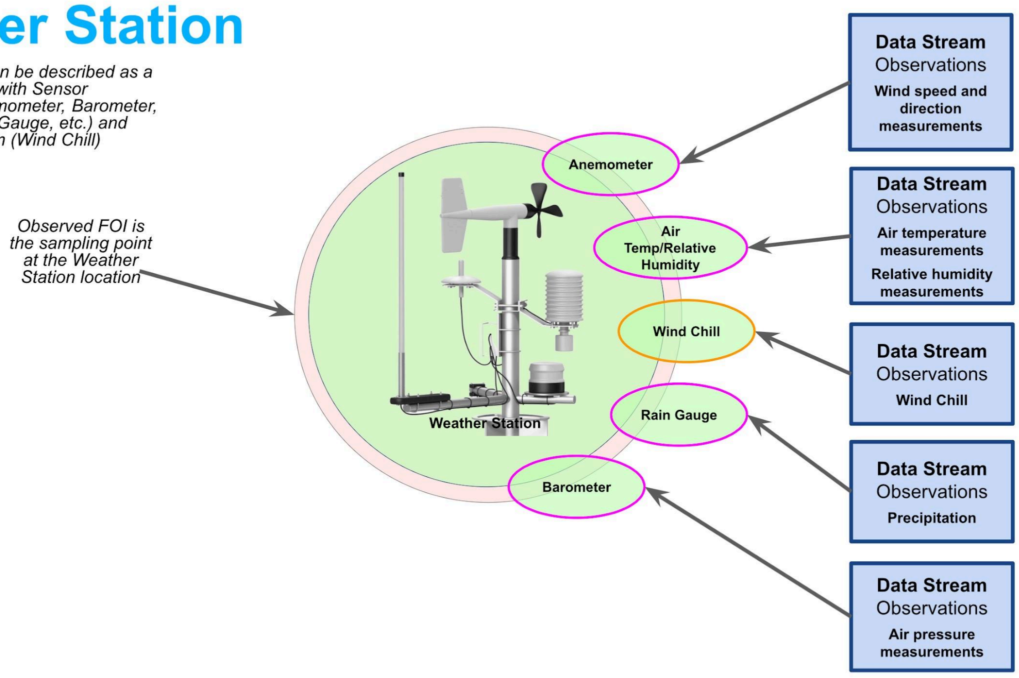
System	**Control Stream**	**Comments**
	None	

7.1.2. Weather Station

While there are simpler sensors (see Thing/IoT above), a Weather Station is a good example of a geographically fixed *in situ* sensing System that collects Observations at a given sampling point. The diagram and discussion below help convey how Weather Stations can be treated in the OGC API - Connected Systems Standard v1.0.

Weather Station

Weather Station can be described as a System (Platform) with Sensor Subsystems (Thermometer, Barometer, Anenometer, Rain Gauge, etc.) and Process Subsystem (Wind Chill)


Observed FOI is the sampling point at the Weather Station location
Anemometer
Air Temp/Relative Humidity
Wind Chill
Rain Gauge
Barometer
Weather Station
Data Stream Observations Wind speed and direction measurements
Data Stream Observations Air temperature measurements Relative humidity measurements
Data Stream Observations Wind Chill
Data Stream Observations Precipitation
Data Stream Observations Air pressure measurements

System: Weather Station (the top level System is the Platform in this case)
Platform: Weather Station (the Platform is the top level System)
Sensors: Thermometer, Barometer, Anemometer, Rain Gauge
Processes: Wind Chill
Actuators: None
Features of Interest: Sampling point at the Weather Station location
Data Streams/Observations: One Data Stream per Sensor/Process (see below)
Control Streams/Commands: Change Sensor configuration (e.g. sampling rate) (one Control Stream per Sensor)

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Weather Station	System	
Weather Station	Platform	the top level System is the Platform in this case
Thermometer	Sensor	Subsystem mounted on the Platform
Barometer	Sensor	Subsystem mounted on the Platform
Anemometer	Sensor	Subsystem mounted on the Platform
Rain Gauge	Sensor	Subsystem mounted on the Platform

Wind Chill	Process	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Sampling point at the Weather Station location	All Sensors	All Sensors measure parameters of the same Feature of Interest

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

System	**Data Stream**	**Comments**
Thermometer	Air temperature and relative humidity measurements	
Barometer	Air pressure measurements	
Anemometer	Wind speed and direction measurements	
Rain Gauge	Precipitation measurements	
Wind Chill Process	Wind chill measurements	Wind chill is calculated from temperature, wind speed, and humidity

Control Streams/Commands:

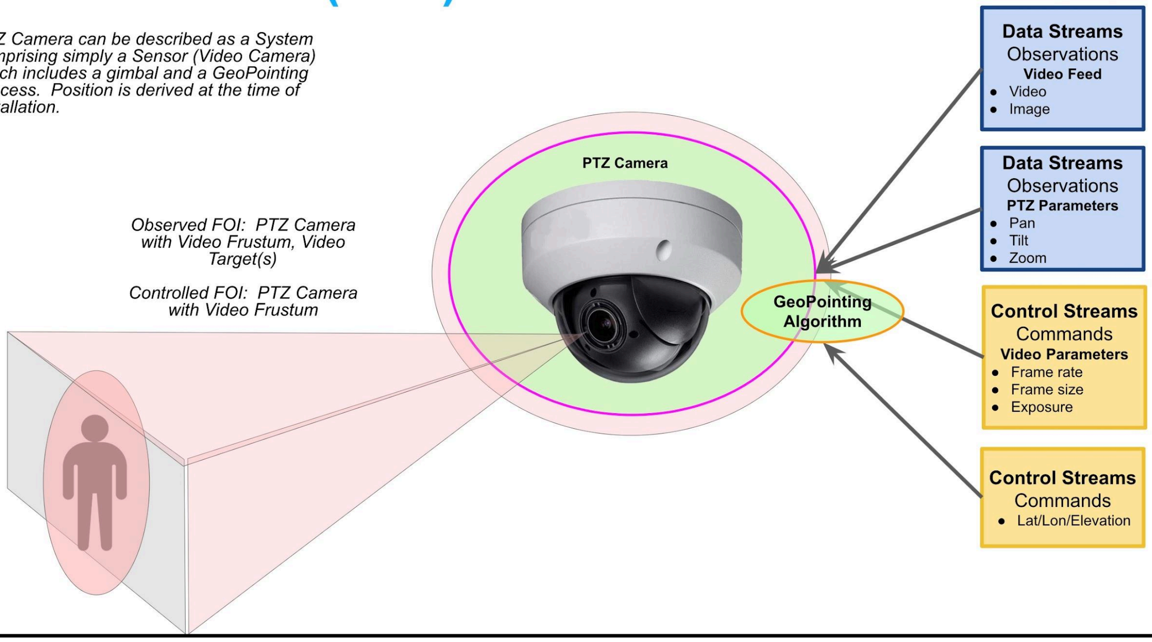
System	Control Stream	Comments
Thermometer	Change sensor config (e.g., sampling rate)	This Control Stream is directly available on the Sensor resource itself (no need for an additional Actuator)
Barometer	Change Sensor config (e.g., sampling rate)	This Control Stream is directly available on the Sensor resource itself (no need for an additional Actuator)
Anemometer	Change Sensor config (e.g., sampling rate)	This Control Stream is directly available on the Sensor resource itself (no need for an additional Actuator)
Rain Gauge	Change Sensor config (e.g., sampling rate)	This Control Stream is directly available on the Sensor resource itself (no need for an additional Actuator)
Wind Chill Process	Change Process config (e.g., sampling rate)	This Control Stream is directly available on the Process resource itself (no need for an additional Actuator)

7.1.3. Pan Tilt Zoom (PTZ) Camera

A PTZ Camera is an example of a fixed sensor that can be tasked to remotely observe its surroundings. In this example, the position and orientation (GeoPose) is configured at the time of installation. While it is possible to have a PTZ Camera that derives GeoPose from GNSS/INS, that is not contemplated in this particular use case. This example is intentionally ‘stripped down’, combining the Actuator of the gimbal within the PTZ Camera description for the purpose of simplicity, since a Control Stream can be used to Command any System, whether primarily Sensor, Process, or Actuator. The diagram and discussion below help convey how PTZ Cameras can be treated in the OGC API - Connected Systems Standard v1.0.

Pan Tilt Zoom (PTZ) Camera

PTZ Camera can be described as a System comprising simply a Sensor (Video Camera) which includes a gimbal and a GeoPointing Process. Position is derived at the time of installation.



System: PTZ Camera
Platform: None
Sensors: PTZ Camera (the Sensor is the top level System)
Actuators: None
Processes: GeoPointing Algorithm
Features of Interest: PTZ Camera, with Frustum, Video Target(s)
Data Streams/Observations: Video, PTZ Parameters
Control Streams/Command: Raw PTZ, Point to Location, Change Video Parameters

Systems:

Name	**Type**	**Description (+ link to datasheet)**
PTZ Camera	System	
PTZ Camera	Sensor	Subsystem mounted on the Platform
GeoPointing Algorithm	Process	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**

PTZ Camera with Frustum	PTZ Camera	The PTZ Camera provides its own orientation relative to earth, as well as imaging parameters like FOV, frame size, frame rate, etc.
Video Target(s)	PTZ Camera	This is the feature the camera is looking at (e.g. a street intersection, a building, a room inside a building, etc.). The video camera provides imagery of the target.

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
PTZ Camera with Frustum	PTZ Camera	The camera system itself can receive commands to move (rotate) itself.

Data Streams/Observations:

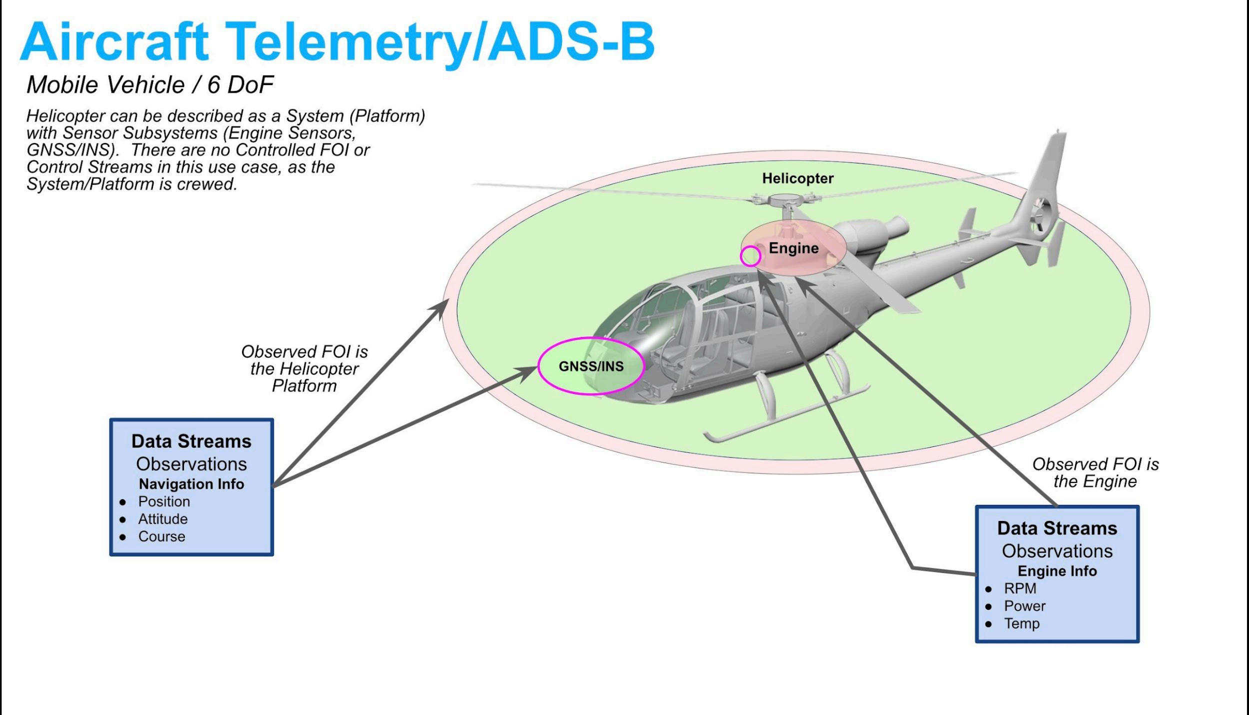
System	**Data Stream**	**Comments**
Video Camera	Video Feed	
Video Camera	PTZ Parameters	

Control Streams/Commands:

System	**Control Stream**	**Comments**
Video Camera	Video Parameters	Change video parameters (e.g. frame rate, frame size, exposure, etc.)
Video Camera	PTZ Parameters	
GeoPointing Algorithm	X,Y,Z,T	Point the PTZ camera to a given lat/lon/elevation

7.1.4. Aircraft Telemetry / ADS-B

Telemetry data from 6 Degree of Freedom (6 DoF) airborne Platforms applies the same to fixed wing aircraft and rotary wing aircraft as it does to missiles and projectiles. Telemetry Sensors generating position, attitude, and course information are critical to deriving the GeoPose of such Platforms, and then, by association, other Sensors on the Platform can derive their own GeoPose information. Given the increasing prevalence of GPS-denial within conflict zones, some military aircraft also come with Assured Position, Navigation, and Timing (A-PNT) solutions that can derive GeoPose information for the aircraft, as a System/Platform, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how aircraft telemetry and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0. For the purposes of this example, there are no Actuators, because we assume the human pilot will control the Platform.



Aircraft Telemetry/ADS-B
Mobile Vehicle / 6 DoF
Helicopter can be described as a System (Platform) with Sensor Subsystems (Engine Sensors, GNSS/INS). There are no Controlled FOI or Control Streams in this use case, as the System/Platform is crewed.
Helicopter
Engine
GNSS/INS
Observed FOI is the Helicopter Platform
Data Streams
Observations
Navigation Info
Position
Attitude
Course
Observed FOI is the Engine
Data Streams
Observations
Engine Info
RPM
Power
Temp

System: Helicopter (the top level System is the Platform in this case)
Platform: Helicopter (the Platform is the top level System)
Sensors: GNSS/INS, Engine Sensors (Subsystems mounted on the Platform)
Actuators: None
Processes: None
Features of Interest: Helicopter, Engine
Data Streams/Observations: Positioning Data, Engine State
Control Streams/Commands: None

Systems:

Name	Type	Description (+ link to datasheet)
Helicopter	System	

Helicopter	Platform	The top level System is the Platform in this case
GNSS/INS	Sensor	Subsystem mounted on the Platform
Engine Sensors	Sensor	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Helicopter	GNSS/INS	GNSS/INS provides position and orientation of the Helicopter Platform
Engine	Engine Sensor	Engine Sensors provide measurements of engine parameters

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
None	None	This example is manned/piloted, therefore there are no controllable parameters in this model.

Data Streams/Observations:

System	**Data Stream**	**Comments**
GNSS/INS	Aircraft Positioning Data	Position, attitude, velocity, acceleration, positioning accuracy, etc.
Engine Sensor	Engine State	Engine parameters (e.g. temp, power, rpm, etc.)

Control Streams/Commands:

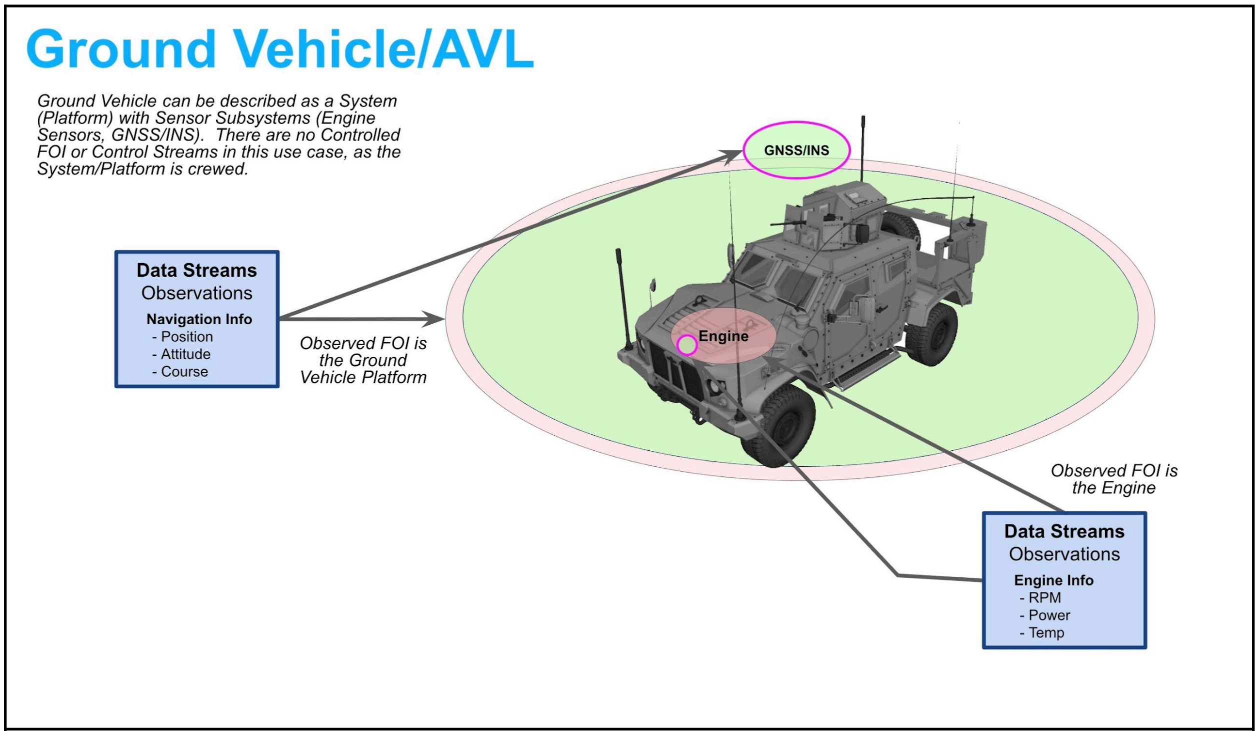
System	**Control Stream**	**Comments**
None	None	

Note: This is a purposefully simple example that could be further enhanced by:

- Adding more Sensors providing state of the Aircraft (e.g. air speed, temp, etc.)
- Adding Control Channels to communicate mission info to the pilot
- Adding one or more payloads, each with its own Data Stream(s) and Control Channel(s)

7.1.5. Ground Vehicle Telemetry / AVL

Ground vehicles increasingly come with sophisticated telemetry Sensors. Given the increasing prevalence of GNSS-denial within conflict zones, some military vehicles also come with Assured Position, Navigation, and Timing (A-PNT) solutions that can derive GeoPose information for the ground vehicle, as a System/Platform, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how ground vehicle telemetry and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0. For the purposes of this example, there are no Actuators, because we assume the human driver will control the Platform.


Ground Vehicle/AVL
Ground Vehicle can be described as a System (Platform) with Sensor Subsystems (Engine Sensors, GNSS/INS). There are no Controlled FOI or Control Streams in this use case, as the System/Platform is crewed.
GNSS/INS
Data Streams
Observations
Navigation Info
- Position
- Attitude
- Course
Observed FOI is the Ground Vehicle Platform
Engine
Observed FOI is the Engine
Data Streams
Observations
Engine Info
- RPM
- Power
- Temp

System: Ground Vehicle (the top level System is the Platform in this case)
Platform: Ground Vehicle (the Platform is the top level System)
Sensors: GNSS/INS, Engine Sensors (Subsystems mounted on the Platform)
Actuators: None
Processes: None
Features of Interest: Ground Vehicle, Engine
Data Streams/Observations: Positioning Data, Engine State
Control Streams/Commands: None

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Ground Vehicle	System	
Ground Vehicle	Platform	the top level System is the Platform in this case
GNSS/INS	Sensor	Subsystem mounted on the Platform
Engine Sensors	Sensor	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Ground Vehicle	GNSS/INS	GNSS/INS provides position and orientation of the vehicle
Engine	Engine Sensor	Engine Sensors provide measurements of engine parameters

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
None	None	This example is manned/crewed therefore there are no controllable parameters in this model.

Data Streams/Observations:

System	**Data Stream**	**Comments**
GNSS/INS	Vehicle Positioning Data	Position, attitude, velocity, acceleration, positioning accuracy, etc.
Engine Sensor	Engine State	Engine parameters (e.g. temp, power, rpm, etc.)

Control Streams/Commands:

System	**Control Stream**	**Comments**

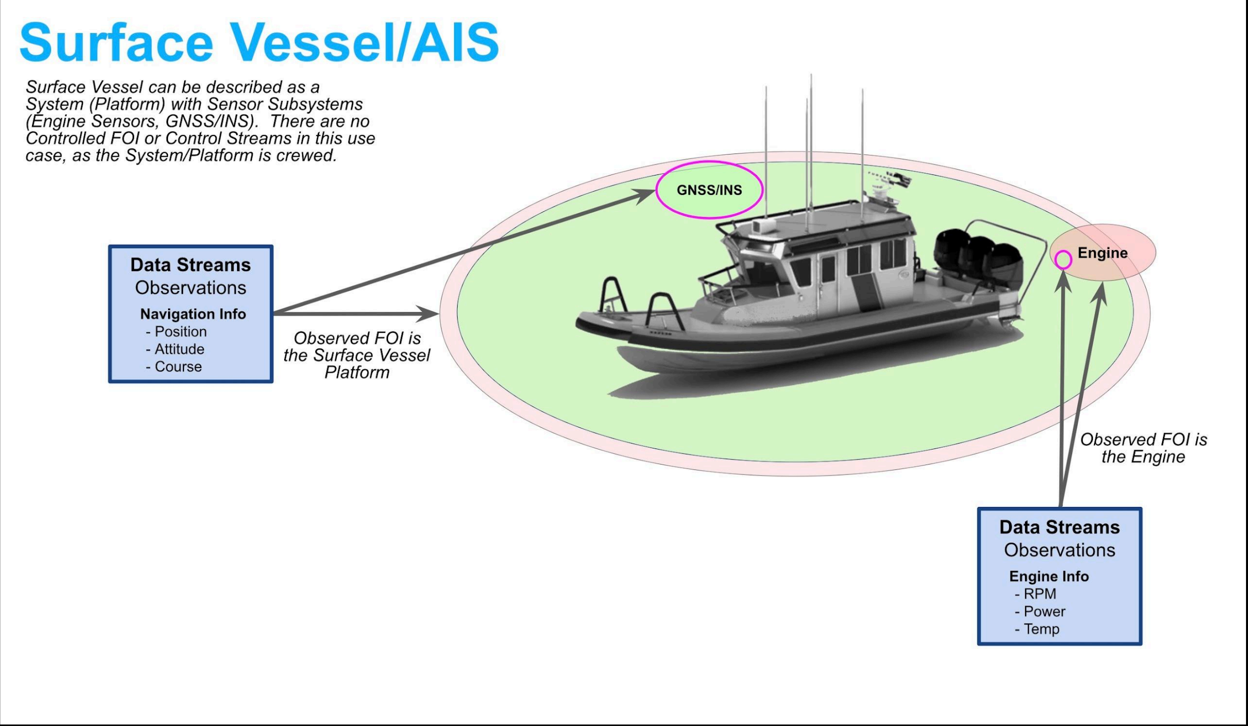
None	None	

Note: This is a purposefully simple example that could be further enhanced by:

- Adding more Sensors providing state of the Ground Vehicle (e.g. ground speed, temp, etc.)
- Adding Control Channels to communicate mission info to the driver
- Adding one or more payloads, each with its own Data Stream(s) and Control Channel(s)

7.1.6. Surface Vessel / AIS

Surface vessels have long benefited from onboard GNSS/INS that provide persistent location and heading information to the navigator. Given the increasing prevalence of GNSS-denial within conflict zones, some military surface vessels also come with Assured Position, Navigation, and Timing (A-PNT) solutions that can derive GeoPose information for the surface vessel, as a System/Platform, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how surface vessel telemetry and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0. For the purposes of this example, there are no Actuators, because we assume the human captain will control the Platform.


Surface Vessel/AIS
Surface Vessel can be described as a System (Platform) with Sensor Subsystems (Engine Sensors, GNSS/INS). There are no Controlled FOI or Control Streams in this use case, as the System/Platform is crewed.
GNSS/INS
Engine
Data Streams
Observations
Navigation Info
- Position
- Attitude
- Course
Observed FOI is the Surface Vessel Platform
Observed FOI is the Engine
Data Streams
Observations
Engine Info
- RPM
- Power
- Temp

System: Surface Vessel (the top level System is the Platform in this case)
Platform: Surface Vessel (the Platform is the top level System)

Sensors: GNSS/INS, Engine Sensors (Subsystems mounted on the Platform)
Actuators: None
Processes: None
Features of Interest: Surface Vessel, Engine
Data Streams/Observations: Positioning Data, Engine Performance
Control Streams/Commands: None

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Surface Vessel	System	
Surface Vessel	Platform	the top level System is the Platform in this case
GNSS/INS	Sensor	Subsystem mounted on the Platform
Engine Sensors	Sensor	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Surface Vessel	GNSS/INS	GNSS/INS provides position and orientation (e.g., ‘orientation at rest’, heal, trim, heading of the Vessel Platform
Engine	Engine Sensor	Engine Sensors provide measurements of Engine parameters

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
None	None	This example is manned/crewed/captained, therefore there are no controllable parameters in this model.

Data Streams/Observations:

System	**Data Stream**	**Comments**

GNSS/INS	Vessel Positioning Data	Position, attitude, velocity, acceleration, positioning accuracy, etc.
Engine Sensor	Engine State	Engine parameters (e.g. temp, power, rpm, etc.)

Control Streams/Commands:

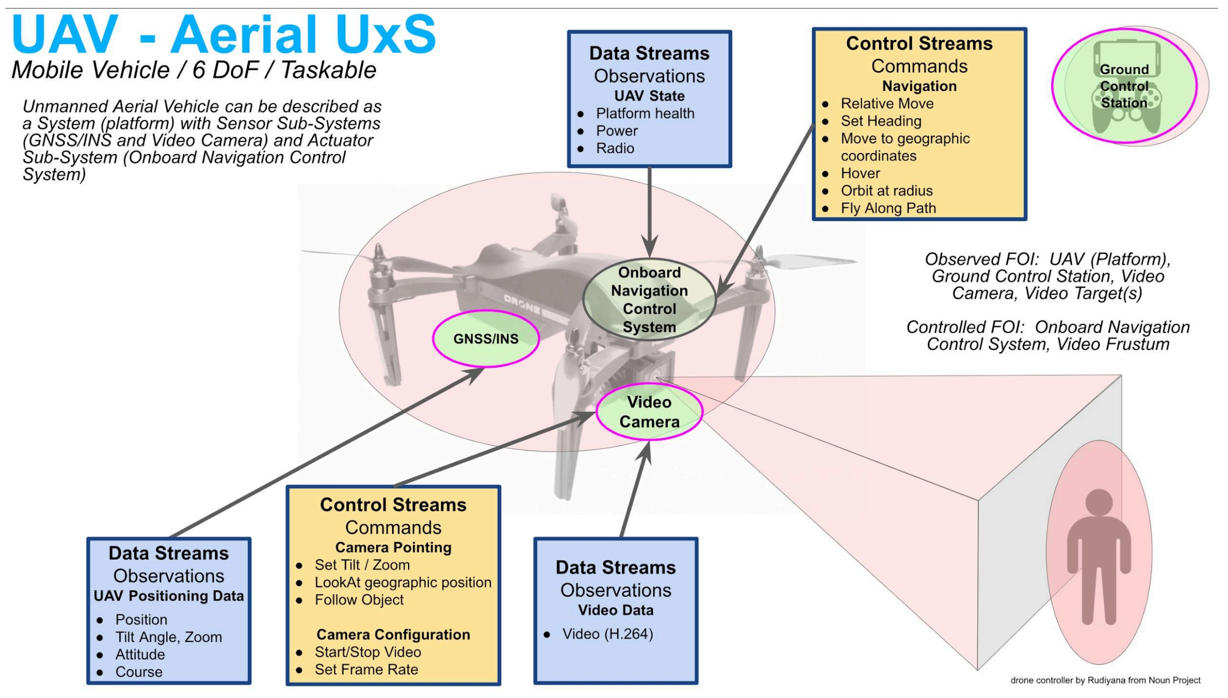
System	**Control Stream**	**Comments**
None	None	

Note: This is a purposefully simple example that could be further enhanced by:

- Adding more sensors providing state of the Surface Vessel (e.g. vessel speed, temp, etc.)
- Adding control channels to communicate mission info to the captain
- Adding one or more payloads, each with its own Data Stream(s) and Control Channel(s)

7.1.7. Unmanned Aerial System (UAS - aka Aerial UxS)

Aerial UxS Platforms increasingly have onboard GNSS/INS that provide persistent 6 DoF GeoPose to the operator or the autonomous navigation process. Given the increasing prevalence of GNSS-denial within conflict zones, aerial UxS producers are increasingly looking to miniaturize Assured Position, Navigation, and Timing (A-PNT) solutions that can derive GeoPose information for the Aerial UxS, as a System/Platform, when GNSS is denied, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how Aerial UxS telemetry and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0.



UAV - Aerial UxS

Mobile Vehicle / 6 DoF / Taskable

Unmanned Aerial Vehicle can be described as a System (platform) with Sensor Sub-Systems (GNSS/INS and Video Camera) and Actuator Sub-System (Onboard Navigation Control System)

Observed FOI: UAV (Platform), Ground Control Station, Video Camera, Video Target(s)

Controlled FOI: Onboard Navigation Control System, Video Frustum

System: Unmanned Aerial System (UAS) (the top level System in this case includes the Platform and the Ground Control Station (GCS))
Platform: Unmanned Aerial Vehicle (UAV)
Sensors: GNSS/INS, Video Camera
Actuators: Onboard Navigation Control System
Processes: GeoPointing Algorithm
Features of Interest: UAV, GCS, Camera Frustum, Video Target(s)
Data Streams/Observations: Positioning Data, Video
Control Streams/Commands: Navigation, Camera pointing, Camera config

Systems:

Name	**Type**	**Description (+ link to datasheet)**
UAS	System	
UAV	Platform	The Platform is the first component of the top level System (UAS)
GNSS/INS	Sensor	Subsystem mounted on the Platform
Video Camera	Sensor	Subsystem mounted on the Platform
Onboard Navigation Control System	Actuator	Subsystem mounted on the Platform, also includes processes but not described here.

GeoPointing Algorithm	Process	Subsystem mounted on the Platform
Ground Control Station	System	The GCS is the second component of the top level System (UAS)

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
UAV	GNSS/INS	GNSS/INS provides position/orientation/velocity of the UAV Platform
Ground Control Station	GCS	GCS reports data about itself (e.g. battery, radio status, position). Not all GCS have GNSS/INS describing their position, but increasingly they do.
Video Camera	Video Camera	Video Camera Subsystem provides its own orientation relative to the Platform, as well as imaging parameters like FOV, frame size, frame rate, etc.
Video Target(s)	Video Camera	Video Camera provides imagery of the target

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
UAV	Onboard Navigation Control System	Control Subsystem receives navigation commands to task the UAV to change position or follow a flight plan.
Video Camera	Video Camera	Video Camera Subsystem receives commands to change the imaging parameters
Video Frustum	Video Camera	Video Camera Subsystem receives commands to change the gimbal and thus the frustum orientation
UAV, Video Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the frustum to a particular 3D location. This is a higher level task that breaks down into lower level commands for maneuvering the UAV and rotating the gimbal.

Data Streams/Observations:

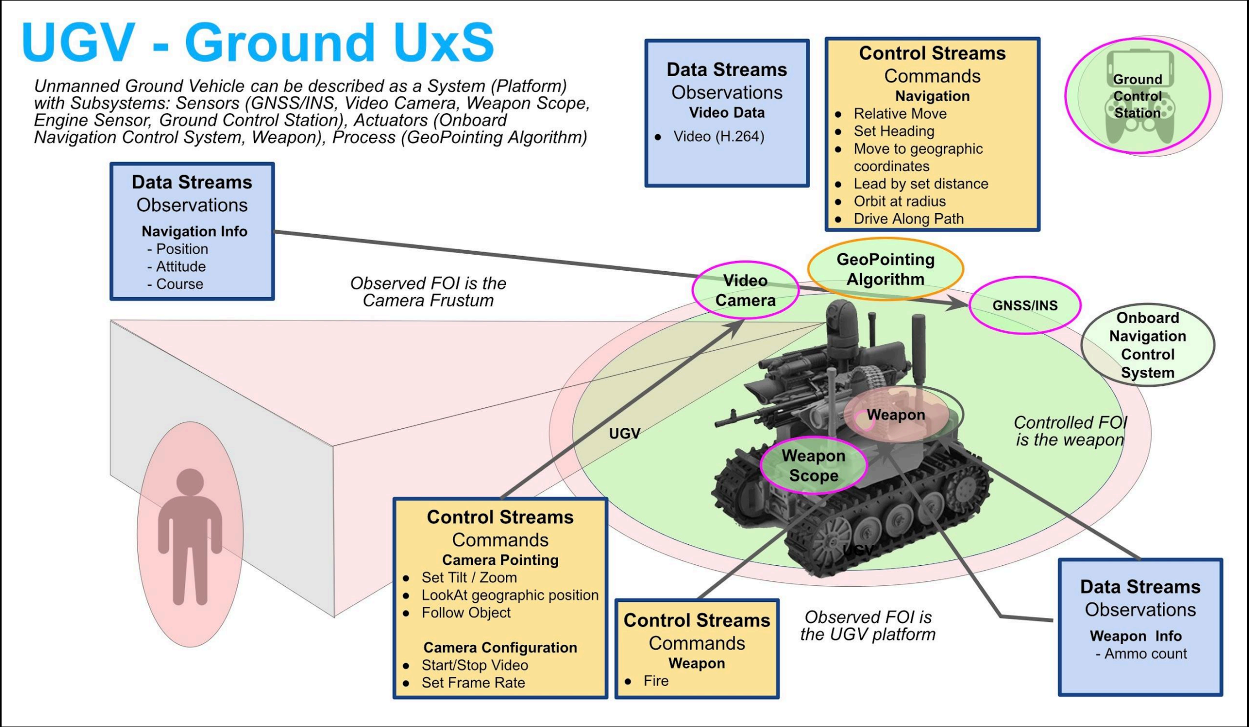
System	Data Stream	Comments
GNSS/INS	UAV Positioning Data	Position, attitude, velocity, acceleration, positioning accuracy, etc.
Onboard Navigation Control System	UAV State	This can include reporting on Platform health, power, radio, etc. details.
Video Camera	Video Data (H.264)	

Control Streams/Commands:

System	Control Stream	Comments
Onboard Navigation Control System	Navigation	- Relative motion (e.g., joystick controls) - Navigate to geographic location - Load and execute entire mission
Video Camera	Camera Pointing	- Raw yaw/pitch/roll command
Video Camera	Camera Configuration	- Start/stop recording - Change frame rate / resolution / exposure, etc.
GeoPointing Algorithm	Camera GeoPointing	Point gimballed camera to X, Y, Z, T

7.1.8. Unmanned Ground Vehicle (UGV - aka Ground UxS)

UGVs increasingly come with sophisticated telemetry Sensors. Given the increasing prevalence of GNSS-denial within conflict zones, some military vehicles also come with Assured Position, Navigation, and Timing (A-PNT) solutions that can derive GeoPose information for the UGV, as a System/Platform, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how UGV telemetry and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0.


UGV - Ground UxS
Unmanned Ground Vehicle can be described as a System (Platform) with Subsystems: Sensors (GNSS/INS, Video Camera, Weapon Scope, Engine Sensor, Ground Control Station), Actuators (Onboard Navigation Control System, Weapon), Process (GeoPointing Algorithm)
Data Streams Observations Video Data • Video (H.264)
Control Streams Commands Navigation • Relative Move • Set Heading • Move to geographic coordinates • Lead by set distance • Orbit at radius • Drive Along Path
Ground Control Station
Data Streams Observations Navigation Info - Position - Attitude - Course
Observed FOI is the Camera Frustum
Video Camera
GeoPointing Algorithm
GNSS/INS
Onboard Navigation Control System
Weapon
Controlled FOI is the weapon
UGV
Weapon Scope
Control Streams Commands Camera Pointing • Set Tilt / Zoom • LookAt geographic position • Follow Object
Camera Configuration • Start/Stop Video • Set Frame Rate
Control Streams Commands Weapon • Fire
Observed FOI is the UGV platform
Data Streams Observations Weapon Info - Ammo count

System: Ground UxS (Unmanned Ground System) (the top level System in this case includes the Platform and the Ground Control Station (GCS))
Platform: UGV (Unmanned Ground Vehicle)
Sensors: GNSS/INS, Video Camera
Actuators: Onboard Navigation Control System, Weapon
Processes: GeoPointing Algorithm
Features of Interest: UGV, Camera Frustum, Object(s)
Data Streams/Observations: Telemetry, Video, Gun scope
Control Streams/Commands: Navigation, Camera pointing, Change camera config (e.g. sampling rate), Gun trigger actuator

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Ground UxS	System	
UGV	Platform	The Platform is the first component of the top level System (UGV)
GNSS/INS	Sensor	Subsystem mounted on the Platform
Video Camera	Sensor	Subsystem mounted on the Platform
Weapon Scope	Sensor	Subsystem mounted on the Platform

Onboard Navigation Control System	Actuator	Subsystem mounted on the Platform, also includes processes but not described here.
GeoPointing Algorithm	Process	Subsystem mounted on the Platform
Weapon	Actuator	Subsystem mounted on the Platform
Ground Control Station	System	The GCS is the second component of the top level System (UGV)

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
UGV	GNSS/INS	GNSS/INS provides position/orientation/velocity of the UGV Platform
Ground Control Station	GCS	GCS reports data about itself (e.g. battery, radio status, position). Not all GCS have GNSS/INS describing their position, but increasingly they do.
Video Camera	Video (H.264)	Video Camera Subsystem receives commands to change the imaging parameters
Video Target(s)	Video Camera	Video Camera provides imagery of the target

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
UGV	Onboard Navigation Control System	Control Subsystem receives navigation commands to task the UGV to change position or follow a mission plan.
Video Camera	Video Camera	Video Camera Subsystem receives commands to change the imaging parameters
Video Frustum	Video Camera	Video Camera Subsystem receives commands to change the gimbal and thus the frustum orientation
Weapon Range	Weapon	Weapon controller receives weapon actuation

		commands.
UGV, Video Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the frustum to a particular 3D location. This is a higher level task that breaks down into lower level commands for maneuvering the UAV and rotating the gimbal.

Data Streams/Observations:

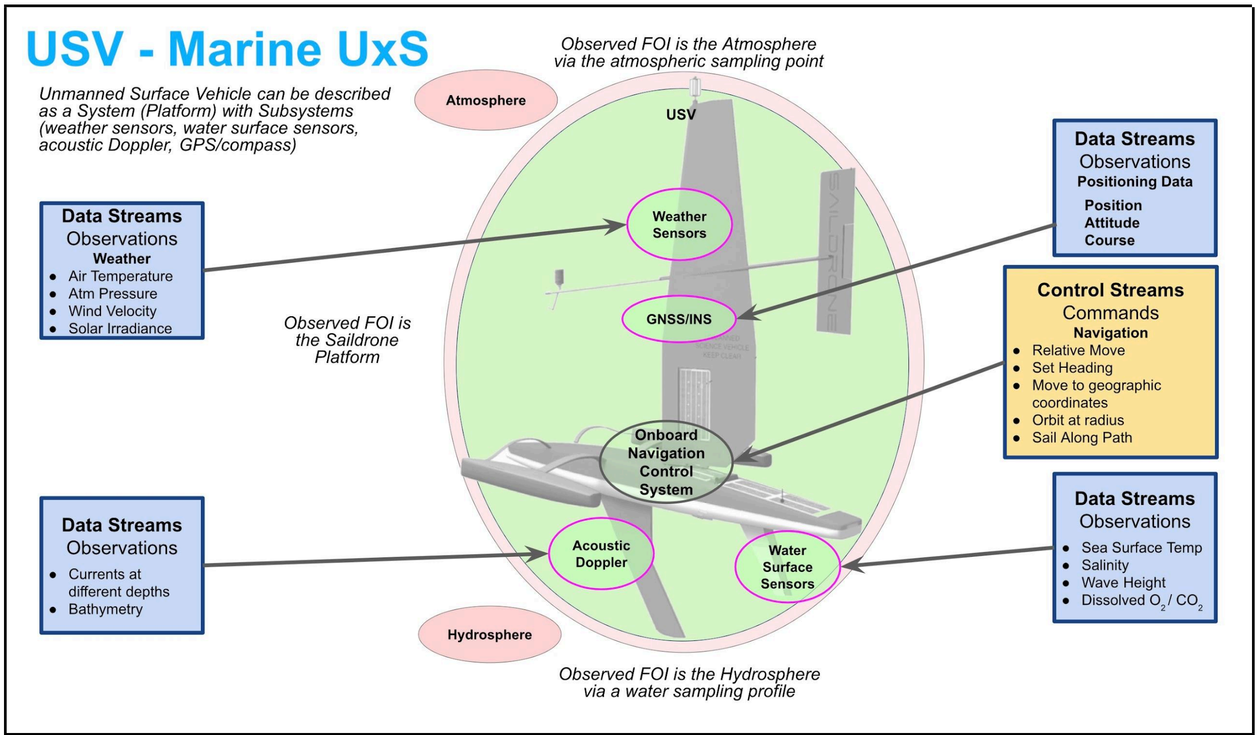
System	**Data Stream**	**Comments**
GNSS/INS	UGV Positioning Data	Position, attitude, velocity, acceleration, positioning accuracy, etc.
Onboard Navigation Control System	UGV State	This can include reporting on Platform health, power, radio, etc. details.
Video Camera	Video (H.264)	

Control Streams/Commands:

System	**Control Stream**	**Comments**
Onboard Navigation Control System	Navigation Commands	- Relative motion (e.g., joystick controls) - Navigate to geographic location - Load and execute entire mission
Video Camera	Camera pointing	- Raw yaw/pitch/roll command
Video Camera	Camera config	- Start/stop recording - Change frame rate / resolution / exposure, etc.
GeoPointing Algorithm	Camera GeoPointing	Point gimballed camera to X, Y, Z, T
Weapon	Weapon Actuation Commands	

7.1.9. Unmanned Surface Vehicles (USV - aka Marine UxS)

USV have long benefited from onboard GPS and magnetic compasses that provide persistent location and heading information to the navigator. Given the increasing prevalence of GPS-denial within conflict zones, some military USV also come with Assured Position, Navigation, and Timing (A-PNT) solutions that can derive GeoPose information for the USV, as a System/Platform, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how USV telemetry and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0.


USV - Marine UxS
Unmanned Surface Vehicle can be described as a System (Platform) with Subsystems (weather sensors, water surface sensors, acoustic Doppler, GPS/compass)
Observed FOI is the Atmosphere via the atmospheric sampling point
Atmosphere
USV
Data Streams Observations Weather
Air Temperature
Atm Pressure
Wind Velocity
Solar Irradiance
Weather Sensors
GNSS/INS
Observed FOI is the Saildrone Platform
Data Streams Observations Positioning Data
Position
Attitude
Course
Control Streams Commands Navigation
Relative Move
Set Heading
Move to geographic coordinates
Orbit at radius
Sail Along Path
Onboard Navigation Control System
Data Streams Observations
Currents at different depths
Bathymetry
Acoustic Doppler
Water Surface Sensors
Data Streams Observations
Sea Surface Temp
Salinity
Wave Height
Dissolved O2 / CO2
Hydrosphere
Observed FOI is the Hydrosphere via a water sampling profile

System: Marine UxS (Unmanned Surface Vehicle)
Platform: USV (Unmanned Surface Vehicle)
Sensors: GNSS/INS, Thermometer/Humidity Probe, Wind Sensor, CDT, Dissolved Oxygen Sensor, Acoustic Doppler, Barometer
Actuators: Onboard Navigation Control System
Processes: None
Features of Interest: USV, Atmosphere, Hydrosphere
Data Streams/Observations: Telemetry, Weather, Acoustic Doppler (Bathymetry, etc.), SST, Salinity, Wave Height, Dissolved O2/CO2
Control Streams/Commands: Navigation, Change sensor config (e.g. sampling rate)

Systems:

Name	**Type**	**Description (+ link to datasheet)**

Marine UxS	System	
USV	Platform	the top level System is the Platform in this case
GNSS/INS	Sensor	Subsystem mounted on the Platform
Weather Sensors	Sensor	Subsystem mounted on the Platform
Water Surface Sensors	Sensor	Subsystem mounted on the Platform (includes SST, Salinity, Wave Height, Dissolved O2/CO2 sensors)
Acoustic Doppler (Sonar)	Sensor	Subsystem mounted on the Platform
Onboard Navigation Control System	Actuator	Subsystem mounted on the Platform, also includes processes but not described here.

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
USV	GNSS/INS	GNSS/INS provides position/orientation/velocity of the USV Platform
Atmosphere	All Weather Sensors	
Hydrosphere	Water Surface Sensors	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
USV	Controller	Controller receives navigation commands to task the USV to change position or follow a flight plan.

Data Streams/Observations:

System	**Data Stream**	**Comments**

GNSS/INS	USV Positioning Data	
Weather Sensors	Air Temperature Atm Pressure Wind Velocity Solar Irradiance	
Water Sensors	Sea Surface Temperature	
Water Sensors	Salinity PPM	
Water Sensors	Wave Height	
Water Sensors	Dissolved O2/CO2	
Acoustic Doppler	Doppler radar of currents, bathymetry, etc.	

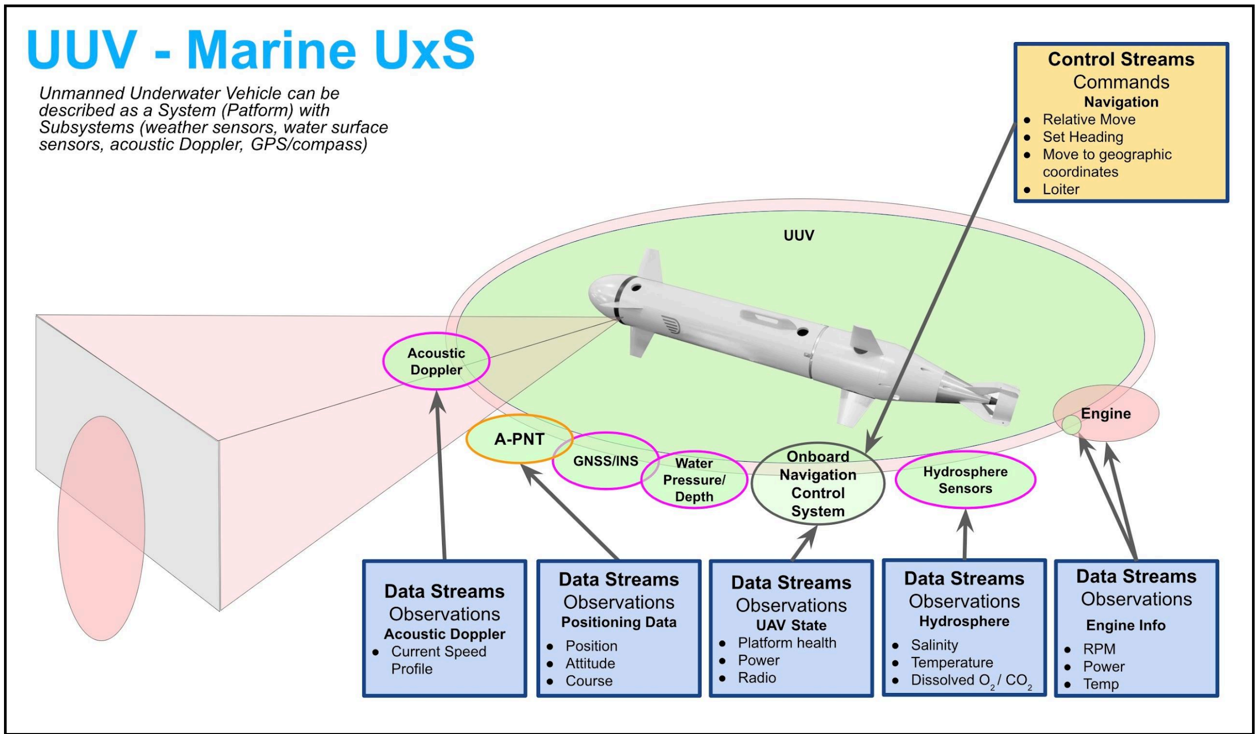
Control Streams/Commands:

System	**Control Stream**	**Comments**
Onboard Navigation Control System		
Sensors		Change sensor sample rate

Note: All weather Sensors are combined into a single System within the Control Streams/Commands section of this presentation for the sake of brevity.

7.1.10. Unmanned Underwater Vehicle (UUV - aka Marine UxS)

UUVs have long benefited from onboard GPS, magnetic compasses, and inertial measurement units (IMU) that provide persistent location and heading information to the navigation system, albeit interpolated between GPS readings. Given the increasing prevalence of GPS-denial within conflict zones, some military UUV also come with Assured Position, Navigation, and Timing (A-PNT) solutions that can derive GeoPose information for the underwater vessel, as a System/Platform, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how UUV telemetry and associated sensors can be treated in the OGC API - Connected Systems Standard v1.0.


UUV - Marine UxS
Unmanned Underwater Vehicle can be described as a System (Patform) with Subsystems (weather sensors, water surface sensors, acoustic Doppler, GPS/compass)
Control Streams
Commands
Navigation
Relative Move
Set Heading
Move to geographic coordinates
Loiter
UUV
Acoustic Doppler
A-PNT
GNSS/INS
Water Pressure/ Depth
Onboard Navigation Control System
Hydrosphere Sensors
Engine
Data Streams Observations Acoustic Doppler
Current Speed Profile
Data Streams Observations Positioning Data
Position
Attitude
Course
Data Streams Observations UAV State
Platform health
Power
Radio
Data Streams Observations Hydrosphere
Salinity
Temperature
Dissolved O2 / CO2
Data Streams Observations Engine Info
RPM
Power
Temp

System: Marine UXS (Unmanned Underwater Vehicle)
Platform: UUV (Unmanned Underwater Vehicle)
Sensors: GNSS/INS, Water Pressure/Depth, Engine, Hydrosphere Sensors, Acoustic Doppler
Actuators: Flight Controller
Processes: A-PNT, GeoPointing Algorithm
Features of Interest: UUV, Sonar Range, Acoustic Doppler (e.g., Sonar) Target(s) [Object(s)], Hydrosphere, Engine
Data Streams/Observations: Positioning Data, Acoustic Doppler (Bathymetry, etc.), Hydrosphere Data
Control Streams/Commands: Navigation, Change sensor config (e.g. sampling rate)

Systems:

Name	Type	Description (+ link to datasheet)
Marine UxS	System	
UUV	Platform	the top level System is the Platform in this case
GNSS/INS	Sensor	Subsystem mounted on the Platform
Water Pressure/Depth	Sensor	Subsystem mounted on the Platform
Hydrosphere Sensors	Sensor (s)	Subsystem mounted on the Platform
Sonar	Sensor	Subsystem mounted on the Platform
Engine	Sensor	Subsystem mounted on the Platform
Onboard Navigation Control System	Actuator	Subsystem mounted on the Platform, also includes processes but not described here.
A-PNT	Process	Subsystem executed on the Platform (GNSS/INS+Water Pressure/Depth)
GeoPointing Algorithm	Process	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
UUV	GNSS/INS + Water Pressure Depth	GNSS/INS and Water Pressure/Depth sensor provides position/orientation/velocity of the UUV Platform
Acoustic Doppler range	Acoustic Doppler	Acoustic Doppler Subsystem provides its own orientation relative to the Platform, as well as sensing parameters like FOV, frame size, frame rate, etc.
Acoustic Doppler Target(s)	Acoustic Doppler	Sonar provides signatures of the target
Engine	Engine Sensor	

Hydrosphere	Hydrosphere Sensors	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
UUV	Controller	Controller receives navigation commands to task the UUV to change position or follow a flight plan.
Sonar	Sonar	Sonar Subsystem receives commands to change the sonar parameters
UUV, Sonar Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the frustum to a particular 3D location. This is a higher level task that breaks down into lower level commands for maneuvering the UUV and rotating the gimbal.

Data Streams/Observations:

System	**Data Stream**	**Comments**
GNSS/INS	UUV Positioning Data	
Sonar		
Hydrosphere Sensors	Hydrosphere Observations	

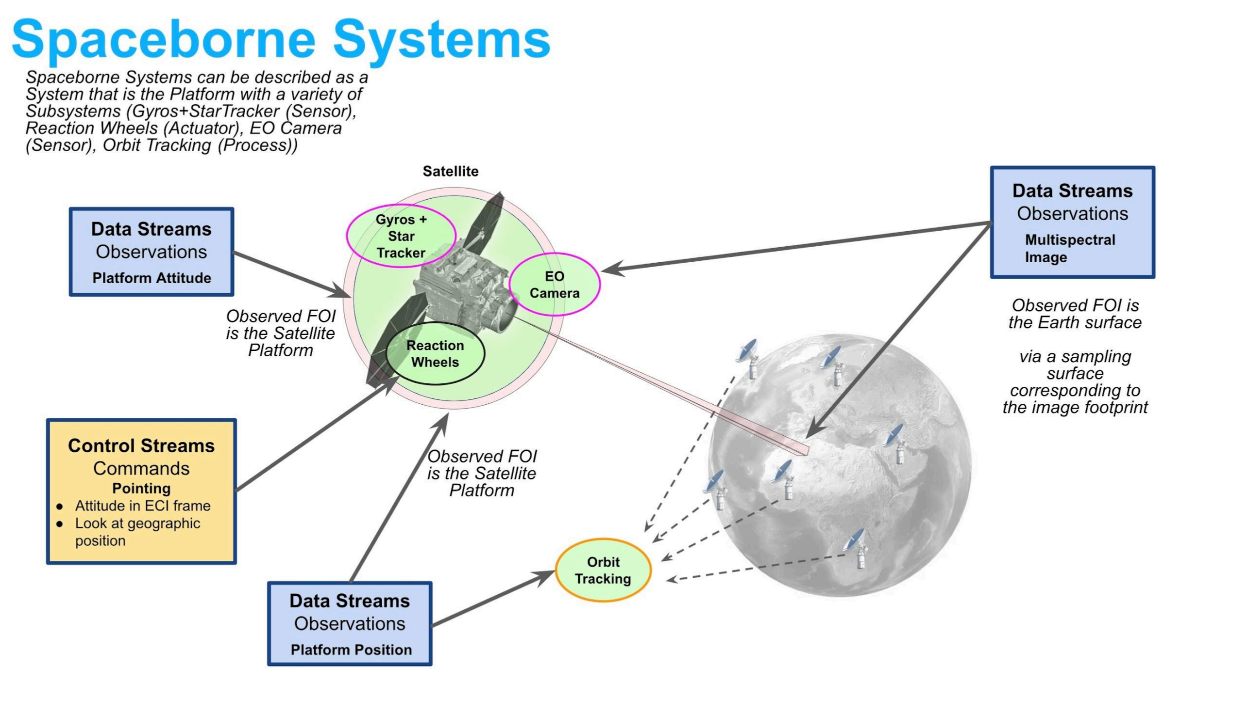
Control Streams/Commands:

System	**Control Stream**	**Comments**
Onboard Navigation Control System	Commands for New Waypoints	
Sonar	Change frequency or sampling ra	

GeoPointing Algorithm	Change the orientation of the UUV	

7.1.11. Spaceborne Systems

Spaceborne Systems, including satellites, have long benefited from onboard GPS, star trackers and other technologies that provide persistent location and orientation information to the Platform navigation system and ground mission control, and by association, for its mounted Sensor Systems. The diagram and discussion below help convey how space borne system telemetry and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0.


Spaceborne Systems
Spaceborne Systems can be described as a System that is the Platform with a variety of Subsystems (Gyros+StarTracker (Sensor), Reaction Wheels (Actuator), EO Camera (Sensor), Orbit Tracking (Process))
Satellite
Data Streams Observations Platform Attitude
Gyros + Star Tracker
EO Camera
Reaction Wheels
Observed FOI is the Satellite Platform
Data Streams Observations Multispectral Image
Observed FOI is the Earth surface
via a sampling surface corresponding to the image footprint
Control Streams Commands Pointing
Attitude in ECI frame
Look at geographic position
Observed FOI is the Satellite Platform
Orbit Tracking
Data Streams Observations Platform Position

System: Space Mission
Platform: Satellite
Sensors: Camera, Startracker, GNSS/INS
Actuators: Gimbal,
Processes: Orbit Tracking, GeoPointing Algorithm
Features of Interest: Satellite, Camera Frustum, Objects
Data Streams/Observations: Images, Video
Control Streams/Commands: Slew to Cue

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Satellite System	System	
Satellite	Platform	the top level System is the Platform in this case
GNSS/INS	Sensor	Subsystem mounted on the Platform
Camera	Sensor	Subsystem mounted on the Platform
Nav Control System	Actuator	Subsystem mounted on the Platform, also includes Processes but not described here.
GeoPointing Algorithm	Process	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Satellite	GNSS/INS	GNSS/INS provides position/orientation/velocity of the UAV Platform
Object(s)	Imager	Imager provides imagery of the target

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
Satellite	Flight Controller	Flight Controller receives navigation commands to task the satellite to change position or follow a flight plan.
Camera	Camera	Camera Subsystem receives commands to change the imaging parameters
Camera Frustum	Camera Gimbal/Bus	Camera Gimbal/Bus Subsystem receives commands to change the Camera Frustum orientation
Satellite, Camera Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the Camera Frustum to a particular 3D location. This is a higher level task that breaks down into lower level

		commands for maneuvering the Satellite and rotating the gimbal or bus.

Data Streams/Observations:

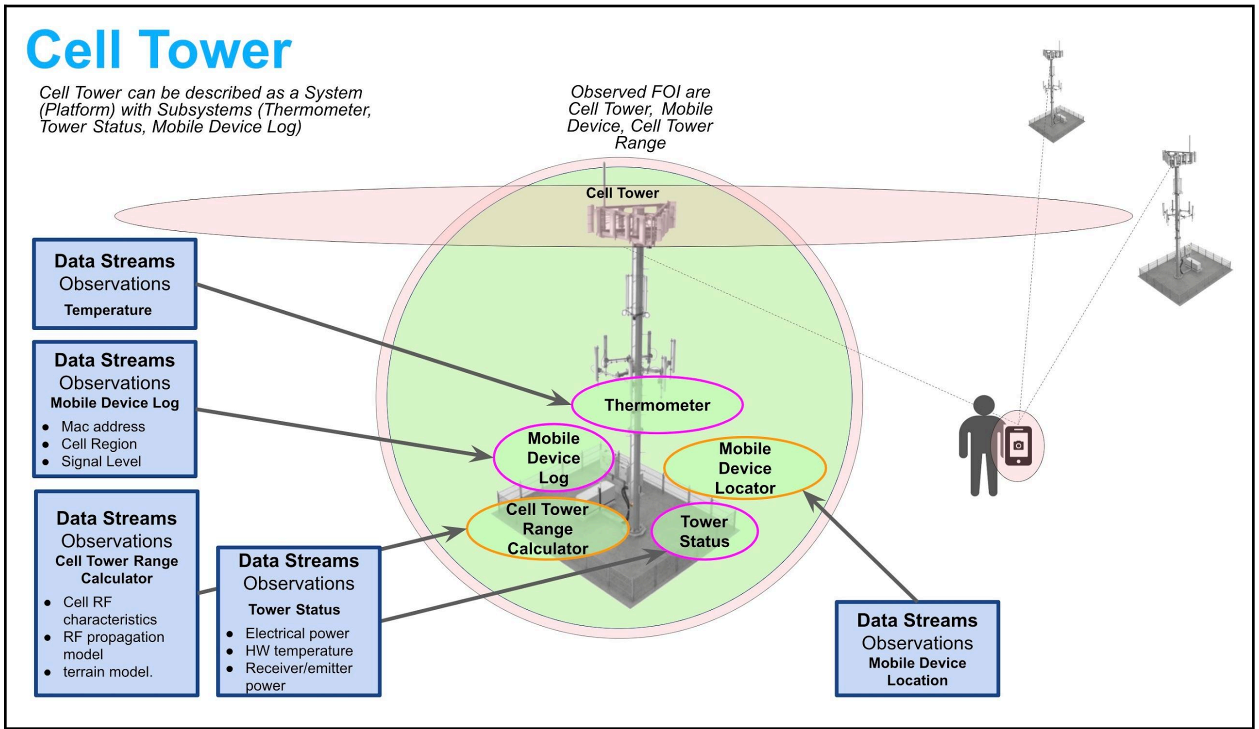
System	Data Stream	Comments
GNSS/INS	Satellite Positioning Data	
Camera	Imagery/Video	

Control Streams/Commands:

System	Control Stream	Comments
Flight Controller		
Camera	Commands to change the imaging parameters	
Imager Gimbal/Bus	Commands to change the Camera Frustum orientation	
GeoPointing Algorithm	Commands to the GeoPointing process to determine where to point the Camera	

7.1.12. Cell Tower

Fixed terrestrial infrastructure such as Cell Towers have long served a key role in helping triangulate the location of mobile handsets in scenarios where the GPS information was not available. The diagram and discussion below help convey how Cell Towers and associated Sensors can be treated in the OGC API - Connected Systems Standard v1.0.


Cell Tower
Cell Tower can be described as a System (Platform) with Subsystems (Thermometer, Tower Status, Mobile Device Log)
Observed FOI are Cell Tower, Mobile Device, Cell Tower Range
Cell Tower
Data Streams Observations Temperature
Data Streams Observations Mobile Device Log
Mac address
Cell Region
Signal Level
Data Streams Observations Cell Tower Range Calculator
Cell RF characteristics
RF propagation model
terrain model.
Data Streams Observations Tower Status
Electrical power
HW temperature
Receiver/emitter power
Thermometer
Mobile Device Log
Mobile Device Locator
Cell Tower Range Calculator
Tower Status
Data Streams Observations Mobile Device Location

System: Cell Tower
Platform: Cell Tower
Sensors: Thermometer, Tower Status Monitor (electrical power, HW temp, receiver/emitter power), Mobile Device Log (mac address, cell region, signal level)
Actuators: None
Processes: Cell Tower Range Calculator, Mobile Device Locator (triangulation)
Features of Interest: Cell Tower, Mobile Device, Cell Tower Range
Data Streams/Observations: One Data Stream per Sensor (see below)
Control Streams/Commands: None

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Cell Tower	System	
Cell Tower	Platform	the top level System is the Platform in this case
Thermometer	Sensor	Subsystem mounted on the Platform
Tower Status Monitor (electrical	Sensor	Subsystem mounted on the Platform

power, HW temp, receiver/emitter power)		
Mobile Device Log (Mac address, Cell Region, Signal Level)	Sensor	Subsystem mounted on the Platform
Cell Tower Range Calculator	Process	Subsystem mounted on the Platform
Mobile Device Locator	Process	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Cell Tower	Tower Status Monitor (Sensor)	electrical power, HW temp, receiver/emitter power
Mobile Device	Mobile Device Log (Sensor)	mac address, cell region, signal level
Cell Tower Range	Cell Tower Range Calculator (Process)	This is calculated with cell RF characteristics, RF propagation model, and terrain model.

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

System	**Data Stream**	**Comments**

Thermometer	Air Temperature	
Tower Status Monitor	Electrical power, HW temp, Receiver/emitter power	
Mobile Device Log	MAC address, Cell region, Signal level	
Cell Tower Range Calculator	Cell Tower Range	
Mobile Device Locator	Mobile Device Location	

Control Streams/Commands:

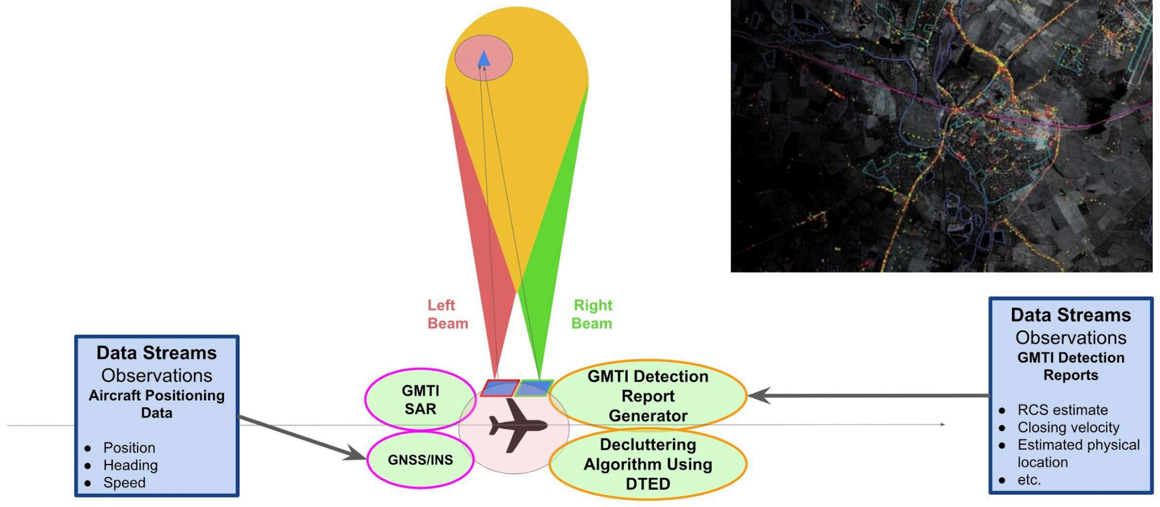
System	**Control Stream**	**Comments**
	None	

7.1.13 GMTI SAR

Ground Moving Target Indicator Synthetic Aperture Radar (GMTI SAR) is an airborne remote sensing capability that discerns Observations of fixed and moving objects on the ground. The diagram and discussion below help convey how GMTI SAR and associated Processes can be treated in the OGC API - Connected Systems Standard v1.0.

GMTI SAR

The Ground Moving Target Indicator Synthetic Aperture Radar (GMTI SAR) comprises a System that is a Sensor, which would be mounted on a Platform that is an Aircraft.



System: GMTI SAR
Platform: Aircraft
Sensors: Synthetic Aperture Radar
Actuators: None
Processes: Decluttering Algorithm Using DTED, GMTI Detection Report Generator
Features of Interest: Aircraft, Left Radar Beam, Right Radar Beam, GMTI Detection
Data Streams/Observations: Raw radar data (echo power at azimuth/elevation) (Left and Right), Decluttered GMTI Detection Reports
Control Streams/Commands: None

Systems:

Name	Type	Description (+ link to datasheet)
GMTI SAR	System	
Aircraft	Platform	the top level System is mounted on the Platform
Radar	Sensor	Subsystem mounted on the Platform
GNSS/INS	Sensor	Subsystem mounted on the Platform
Decluttering Algorithm Using DTED	Process	Subsystem mounted on the Platform
GMTI Detection	Process	Subsystem mounted on the Platform

Report Generator		

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Aircraft	GNSS/INS	
Radar Beams (Left Beam, Right Beam)	Radar	
GMTI Detections	Decluttering Algorithm Using DTED, GMTI Detection Report Generator	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

System	**Data Stream**	**Comments**
Radar	Radar returns	
GMTI Detection Report Generator	GMTI Detections	

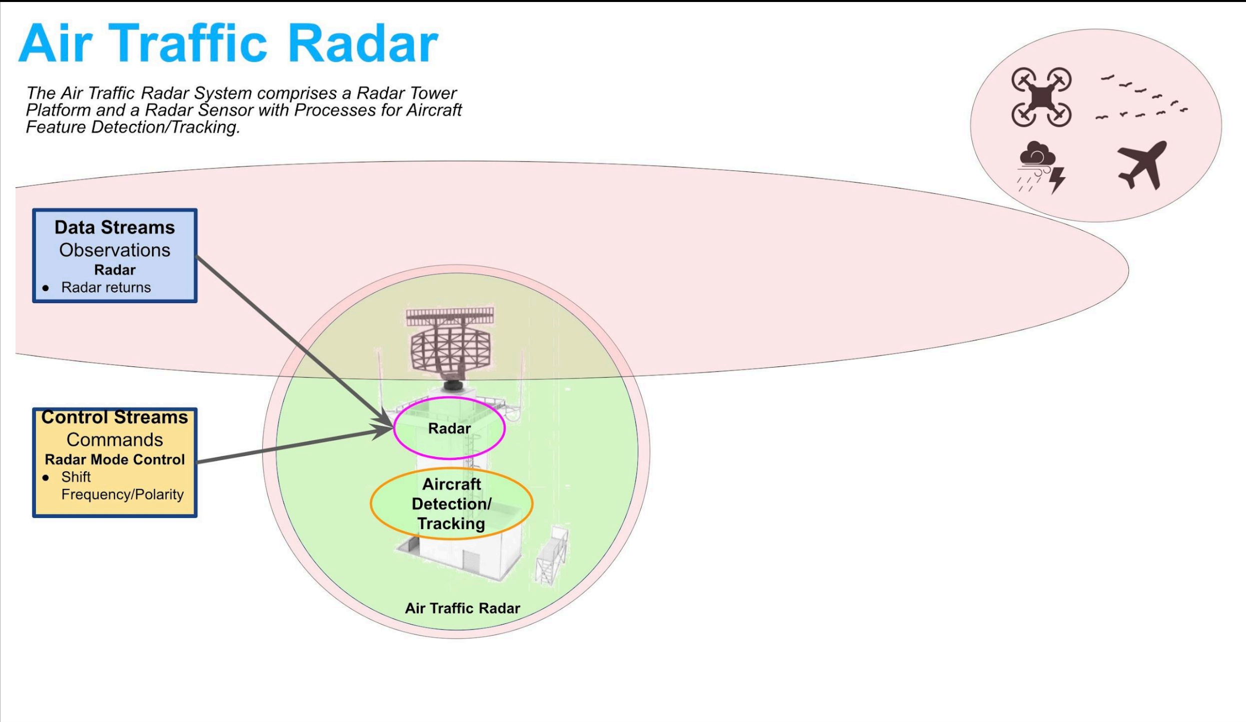
Control Streams/Commands:

System	**Control Stream**	**Comments**

	None	

7.1.14 Air Traffic Radar

Air Traffic Radar is another remote sensing capability tied to a fixed location. Rather than being slewed as a PTZ Camera is, it can be tasked to collect in different modes. The diagram and discussion below help convey how Air Traffic Radar can be treated in the OGC API - Connected Systems Standard v1.0.


Air Traffic Radar
The Air Traffic Radar System comprises a Radar Tower Platform and a Radar Sensor with Processes for Aircraft Feature Detection/Tracking.
Data Streams
Observations
Radar
• Radar returns
Control Streams
Commands
Radar Mode Control
• Shift Frequency/Polarity
Radar
Aircraft Detection/ Tracking
Air Traffic Radar

System: Air Traffic Radar
Platform: Radar Tower
Sensors: Radar
Actuators: None
Processes: Aircraft Tracking
Features of Interest: Radar, Radar Lobes, Targets (aircrafts / drones / birds / weather)
Data Streams/Observations: Raw radar data (echo power at azimuth/elevation), Processed data = aircraft locations
Control Streams/Commands: Radar Mode (shift frequency/polarity, elevation/azimuth speed/increments)

Systems:

Name	Type	Description (+ link to datasheet)
Air Traffic Radar System	System	
Radar Tower	Platform	the top level System is the Platform in this case
Radar	Sensor	Subsystem mounted on the Platform
Aircraft Feature Detection/ Tracking	Process	Subsystem not mounted on Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	System	Comments
Radar	Radar	
Radar Lobes	Radar	
Aircraft	Radar	

Controlled FOI *(the thing you want to control)*	System	Comments
Radar Lobes	RF modulator	
Radar, Radar Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the frustum to a particular 3D location. This is a higher level task that breaks down into lower level commands for pointing the radar.

Data Streams/Observations:

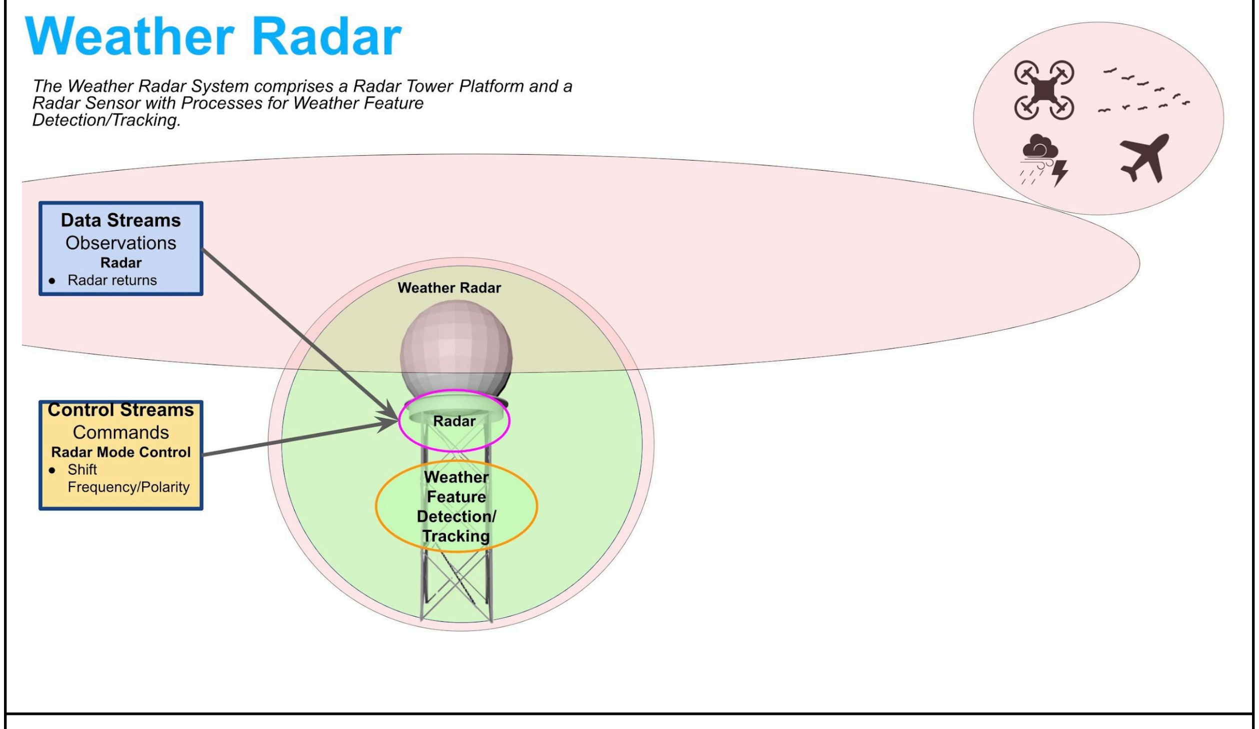
System	Data Stream	Comments
Radar	Radar returns	
Aircraft Feature Detection/ Tracking Process	Aircraft Features	

Control Streams/Commands:

System	Control Stream	Comments
Radar Mode Control	Command a change to the radar modulation	

7.1.15 Weather Radar

Weather Radar is another remote sensing capability tied to a fixed location. Rather than being slewed as a PTZ camera is, it can be tasked to collect in different modes. The diagram and discussion below help convey how Weather Radar can be treated in the OGC API - Connected Systems Standard v1.0.


Weather Radar
The Weather Radar System comprises a Radar Tower Platform and a Radar Sensor with Processes for Weather Feature Detection/Tracking.
Data Streams
Observations
Radar
• Radar returns
Control Streams
Commands
Radar Mode Control
• Shift Frequency/Polarity
Weather Radar
Radar
Weather Feature Detection/ Tracking

System: Weather Radar System
Platform: Radar Tower
Sensors: Radar
Actuators: None

Processes: Weather Feature Detection/Tracking
Features of Interest: Radar, Radar Lobes, Targets (aircrafts / drones / birds / weather)
Data Streams/Observations: Raw radar data (echo power at azimuth/elevation), Processed data = reflectivity (etc.) coverage, additional processing to get features
Control Streams/Commands: Radar Mode (shift frequency/polarity, elevation/azimuth speed/increments)

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Weather Radar System	System	
Radar Tower	Platform	the top level System is the Platform in this case
Radar	Sensor	Subsystem mounted on the Platform
Weather Feature Detection/ Tracking	Process	Subsystem not mounted on Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Radar	Radar	
Radar Lobes	Radar	
Weather System	Radar	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
Radar Lobes	RF modulator	
Radar, Radar Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the frustum to a particular 3D location. This is a higher level task that breaks down into lower level commands for pointing the radar.

Data Streams/Observations:

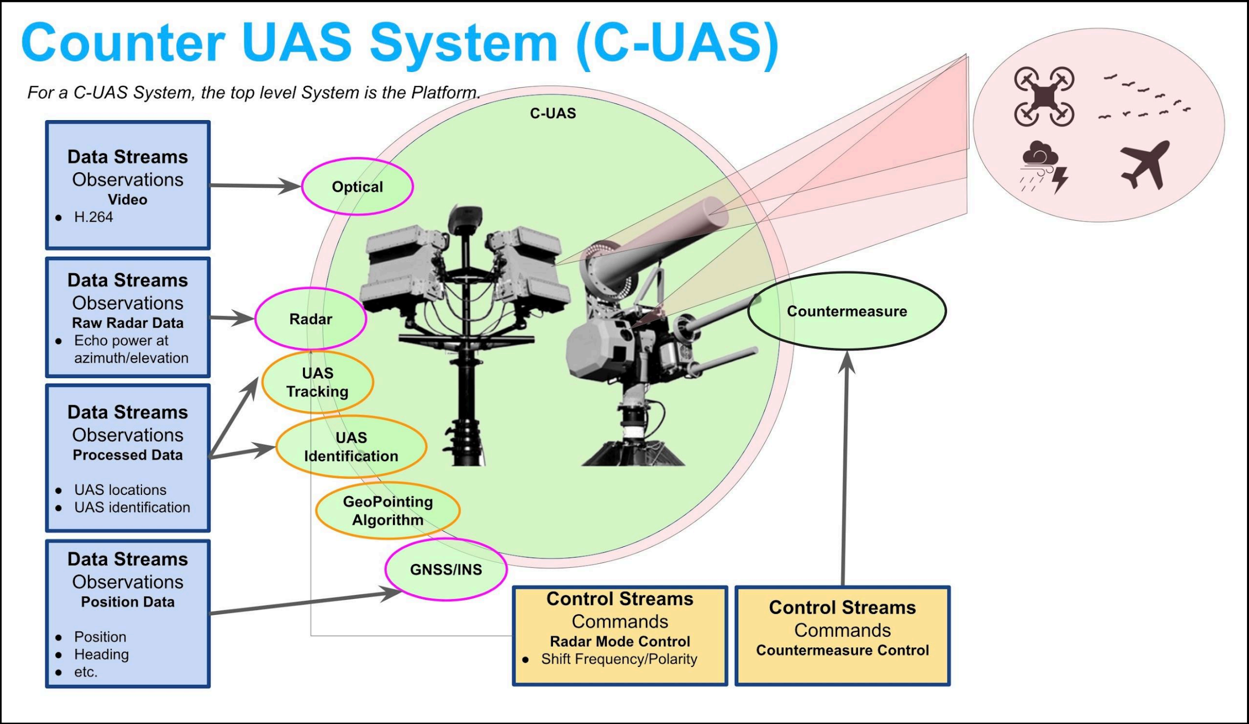
System	Data Stream	Comments
Radar	Radar returns	
Weather Feature Detection/ Tracking Process	Weather Features	

Control Streams/Commands:

System	Control Stream	Comments
Radar Mode Control	Command a change to the radar modulation	

7.1.16. Counter UAS System (C-UAS)

A C-UAS System is an example of a complex System comprising multiple Sensors, Processes, and Actuators which observes and discriminates between Features of Interest of different kinds, and takes action with geospatial precisions and accuracy. A C-UAS System may operate in a fixed location, or while on the move. The diagram and discussion below help convey how C-UAS can be treated in the OGC API - Connected Systems Standard v1.0.



System: C-UAS System
Platform: C-UAS System
Sensors: Radar, Optical, Acoustic,
Actuators: Countermeasures like High energy RF
Processes: UAS Tracking, UAS Identification, GeoPointing Algorithm
Features of Interest: C-UAS System, Radar, Radar Lobes, Countermeasures, Targets (aircrafts / drones / birds / weather)
Data Streams/Observations: Raw radar data (echo power at azimuth/elevation), Processed data = aircraft locations, optical, etc.
Control Streams/Commands: Radar Mode (shift frequency/polarity, elevation/azimuth speed/increments), Countermeasure Command

Systems:

Name	Type	Description (+ link to datasheet)
C-UAS System	Platform	the top level System is the Platform in this case
Radar	Sensor	Subsystem mounted on the Platform
Optical	Sensor	Subsystem mounted on the Platform
Acoustic	Sensor	Subsystem mounted on the Platform
GNSS/INS	Sensor	Subsystem mounted on the Platform

UAS Identification	Process	
UAS Tracking	Process	
GeoPointing Algorithm	Process	
Countermeasures	Actuator	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
C-UAS System	GNSS/INS	GNSS/INS provides position/orientation/velocity of the C-UAS Platform
Radar		Video Camera Subsystem provides its own orientation relative to the Platform, as well as imaging parameters like FOV, frame size, frame rate, etc.
Radar Lobes		
Countermeasures		
Countermeasure Frustum		
Target(s)	Video Camera	Video Camera provides imagery of the target

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
Radar		Flight Controller receives navigation commands to task the UAV to change position or follow a flight plan.
Radar Lobes	Video Camera	Video Camera Subsystem receives commands to change the imaging parameters

Countermeasures	Video Camera	Video Camera Subsystem receives commands to change the gimbal and thus the frustum orientation
Countermeasure Frustum		
C-UAS, Video Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the frustum to a particular 3D location. This is a higher level task that breaks down into lower level commands for maneuvering the CUAS and rotating the gimbal.

Data Streams/Observations:

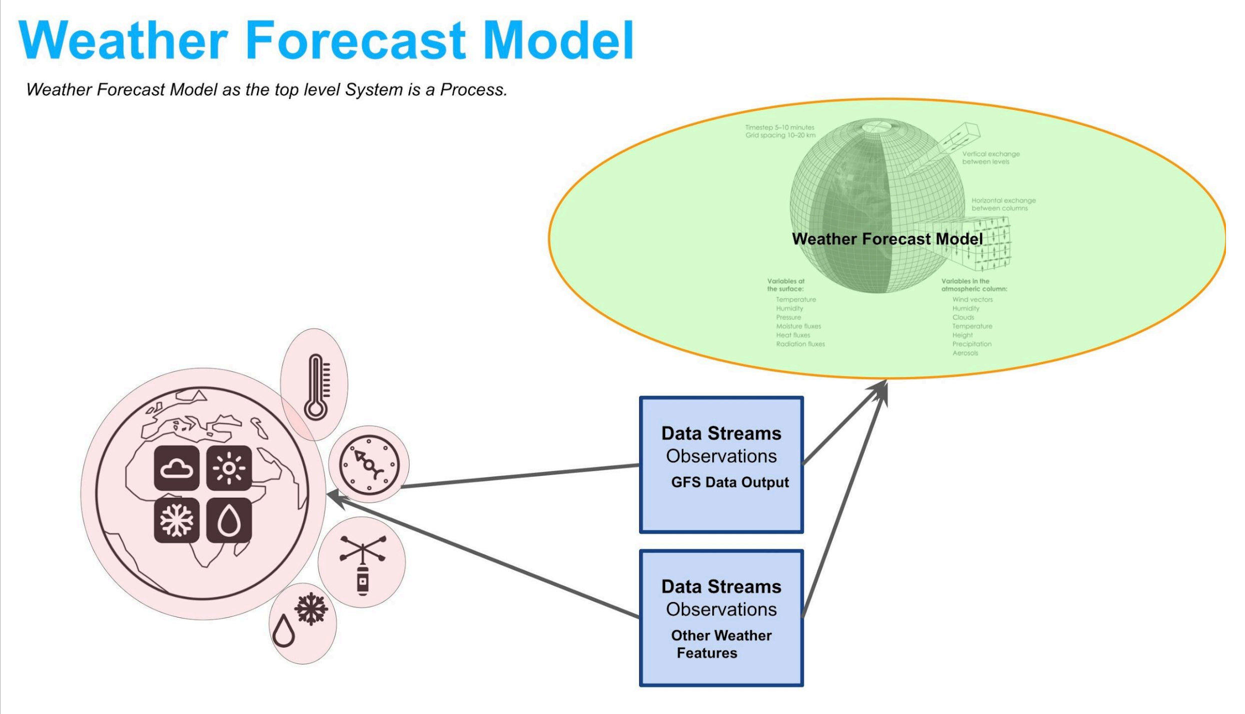
System	**Data Stream**	**Comments**
GNSS/INS	C-UAS Positioning Data	
Radar	Raw Radar Data	
Optical	Video	
UAS Tracking (Process)	Processed Track Data	

Control Streams/Commands:

System	**Control Stream**	**Comments**
Radar Mode Control	Command a change to the radar modulation	
Countermeasure Control	GeoPoint Countermeasure, and Execute	

7.1.17. Weather Forecast Model

A Weather Forecast Model is a System that is a Process which consumes a variety of weather related Sensor feeds, and generates Data Streams of Observations about a variety of Features of Interest comprising our forecasted understanding of global weather. The diagram and discussion below help convey how Weather Forecast Models can be treated in the OGC API - Connected Systems Standard v1.0.



System: Top level System is a Process
Platform: None
Sensors: None
Actuators: None
Processes: The weather forecast model.
Features of Interest: the atmosphere globally (GFS, other weather features can be extracted with additional processing (e.g., temperature, pressure, wind speed, precipitation, etc.)
Data Streams/Observations: 3D grid of the state of the atmosphere at a given time, and predicted at a given time (result time (the time the forecast was run), and phenomenon time (the time you are predicting for)
Control Streams/Commands: None. It runs on its own.

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Weather Forecast Model	System	the top level System is the Process in this case

Weather Forecast Model	Process	the top level System is the Process in this case

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
The atmosphere globally	Process	
Other weather features (e.g., temperature, pressure, wind speed, precipitation, etc.)	Process	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

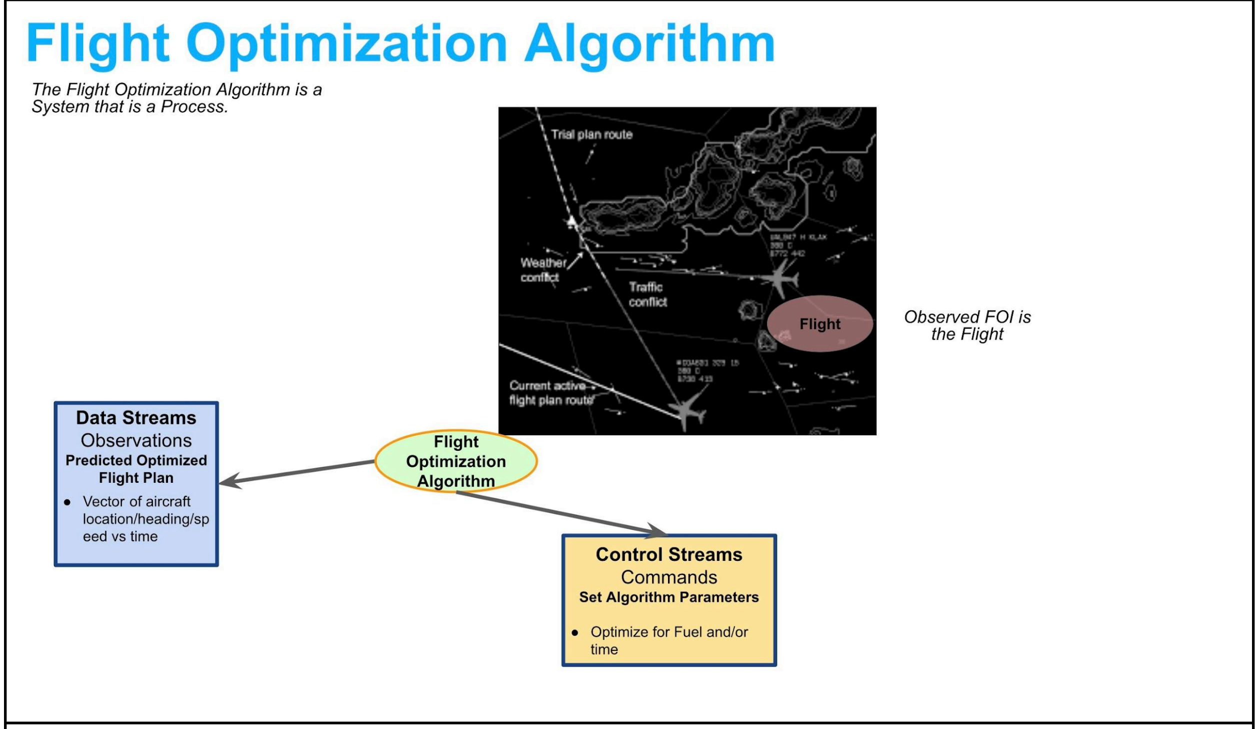
System	**Data Stream**	**Comments**
GFS model	GFS data output	
Additional processes	Other weather feature	

Control Streams/Commands:

System	**Control Stream**	**Comments**
	None	

7.1.18. Flight Optimization Algorithm

A Flight Optimization Algorithm is a System that is a Process which consumes a variety of space-based, airborne and terrestrial weather, aircraft, and flight plan data, and generates Data Streams of Observations that represent optimal flight plans for pilots to choose from. The diagram and discussion below help convey how Flight Optimization Algorithms can be treated in the OGC API - Connected Systems Standard v1.0.


Flight Optimization Algorithm
The Flight Optimization Algorithm is a System that is a Process.
Trial plan route
Weather conflict
Traffic conflict
Current active flight plan route
Flight
Observed FOI is the Flight
Data Streams
Observations
Predicted Optimized Flight Plan
• Vector of aircraft location/heading/speed vs time
Flight Optimization Algorithm
Control Streams
Commands
Set Algorithm Parameters
• Optimize for Fuel and/or time

System: Top level System is a Process.
Platform: None
Sensors: None
Actuators: None
Processes: The Process is the flight optimization algorithm
Features of Interest: The Flight. Different Feature of Interest for each flight number.
Data Streams/Observations: Output of Process is the predicted optimized flight plan (vector of aircraft location/heading/speed vs time). Plan of what the airplane would need to do to fly in an optimal way, given the predicted constraints.
Control Streams/Commands: Set algorithm parameters for given flight ID (e.g. optimize for fuel and/or time); Trigger optimization on-demand (if not automatically triggered); In general, this would run on its own. But, it could also be triggered by a pilot/navigator at any time using an Execute Command, with some parameters beyond Flight ID.

Systems:

Name	Type	Description (+ link to datasheet)
Flight Optimization Algorithm	System	
Flight Optimization Algorithm	Process	the top level System is the Platform in this case

Features of Interest:

Observed FOI *(the thing you want to observe)*	System	Comments
The Flight		GNSS/INS provides position/orientation/velocity of the flight Platform

Controlled FOI *(the thing you want to control)*	System	Comments
	None	

Data Streams/Observations:

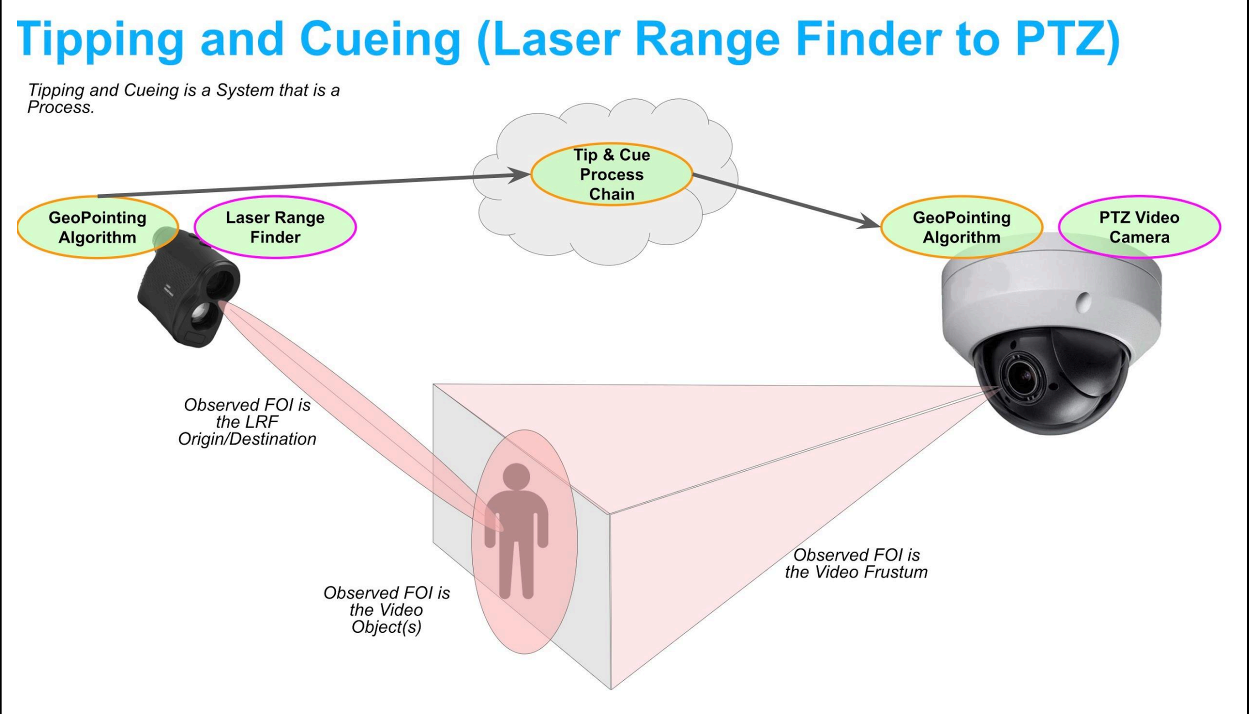
System	Data Stream	Comments
Flight optimization algorithm	Flight optimization	

Control Streams/Commands:

System	Control Stream	Comments
	None	

7.1.19. Tipping and Cueing (Laser Range Finder to PTZ)

A Tipping and Cueing is a System that is a Process which consumes X,Y,Z,T coordinates from one Sensor (e.g., Laser Range Finder - LRF) and forwards them to another Sensor (e.g., PTZ Camera) for the purposes of tasking. The diagram and discussion below help convey how Tipping and Cueing Processes can be treated in the OGC API - Connected Systems Standard v1.0.


Tipping and Cueing (Laser Range Finder to PTZ)
Tipping and Cueing is a System that is a Process.
Tip & Cue Process Chain
GeoPointing Algorithm
Laser Range Finder
GeoPointing Algorithm
PTZ Video Camera
Observed FOI is the LRF Origin/Destination
Observed FOI is the Video Object(s)
Observed FOI is the Video Frustum

System: Top level System is a Process Chain, with Sensor, Processing and Actuator components
Platform: None
Sensors: LRF, PTZ Camera
Actuators: PTZ gimbal, Camera config
Processes: Process is a Tip and Cue of PTZ from Laser Range Finder geolocation.
Features of Interest: Target pointed by LRF in X/Y/Z/T
Data Streams/Observations: None, this is a closed loop Process that sends output data as a Command to an actuator.
Control Streams/Commands: None, this is a closed loop Process that gets inputs directly from Sensors.

Systems:

Name	Type	Description (+ link to datasheet)
Tipping and Cueing	System	

Process Chain		
Tipping and Cueing Process Chain	Process	the top level System is the Process in this case
LRF	Sensor	Subsystem not mounted on the Process
PTZ Camera	Sensor	Subsystem not mounted on the Process
PTZ Gimbal	Actuator	Subsystem not mounted on the Process

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
PTZ Video Frustum	PTZ	Video Camera Subsystem provides its own orientation relative to the Platform, as well as imaging parameters like FOV, frame size, frame rate, etc.
LRF Line of Sight	LRF	This assumes the LRF has GPS, magnetic compass, and accelerometers.
Video Target(s)	Video Camera	Video Camera provides imagery/video of the target

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
Video Camera	Video Camera	Video Camera Subsystem receives commands to change the imaging parameters
Video Frustum	Video Camera	Video Camera Subsystem receives commands to change the gimbal and thus the frustum orientation
PTZ Camera, Video Frustum	GeoPointing Algorithm	GeoPointing process receives commands to point the frustum to a particular 3D location, generated by the LRF. This is a higher level task that breaks down into lower level commands for maneuvering the PTZ and rotating the gimbal.

Data Streams/Observations:

System	**Data Stream**	**Comments**

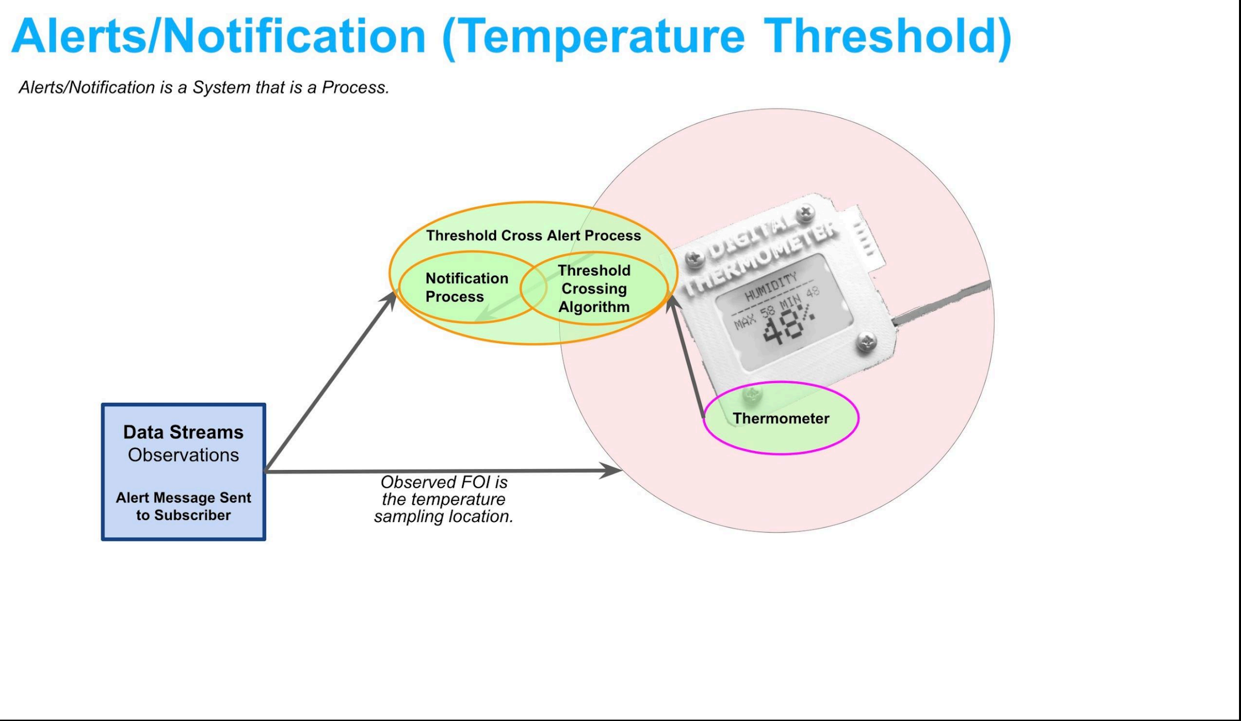
	None	None, this is a closed loop Process, that sends output data as a Command to an actuator.

Control Streams/Commands:

System	Control Stream	Comments
	None	None, this is a closed loop Process that gets inputs directly from Sensors.

7.1.20. Alerts/Notification (Temperature Threshold)

An Alert is a System that is a Process that (in this use case) notifies particular subscribers when a threshold is exceeded. The diagram and discussion below help convey how Alerts/Notification can be treated in the OGC API - Connected Systems Standard v1.0.


Alerts/Notification (Temperature Threshold)
Alerts/Notification is a System that is a Process.
Threshold Cross Alert Process
Notification Process
Threshold Crossing Algorithm
DIGITAL THERMOMETER
Thermometer
Data Streams
Observations
Alert Message Sent to Subscriber
Observed FOI is the temperature sampling location.

System: Top level System is a Process that is fed with data from a Sensor
Platform: None
Sensors: Thermometer

Actuators: None
Processes: Threshold Cross Alert Process
Features of Interest: Temperature sampling location
Data Streams/Observations: Alert message sent to subscriber. (In the API, alerts are just Observations with a different meaning)
Control Streams/Commands: None.

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Alerts/Notification	System	
Alerts/Notification	Process	the top level System is the Platform in this case
Thermometer	Sensor	Subsystem mounted on the Platform
Threshold crossing algorithm	Process	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Temperature sampling location	Temperature Sensor	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

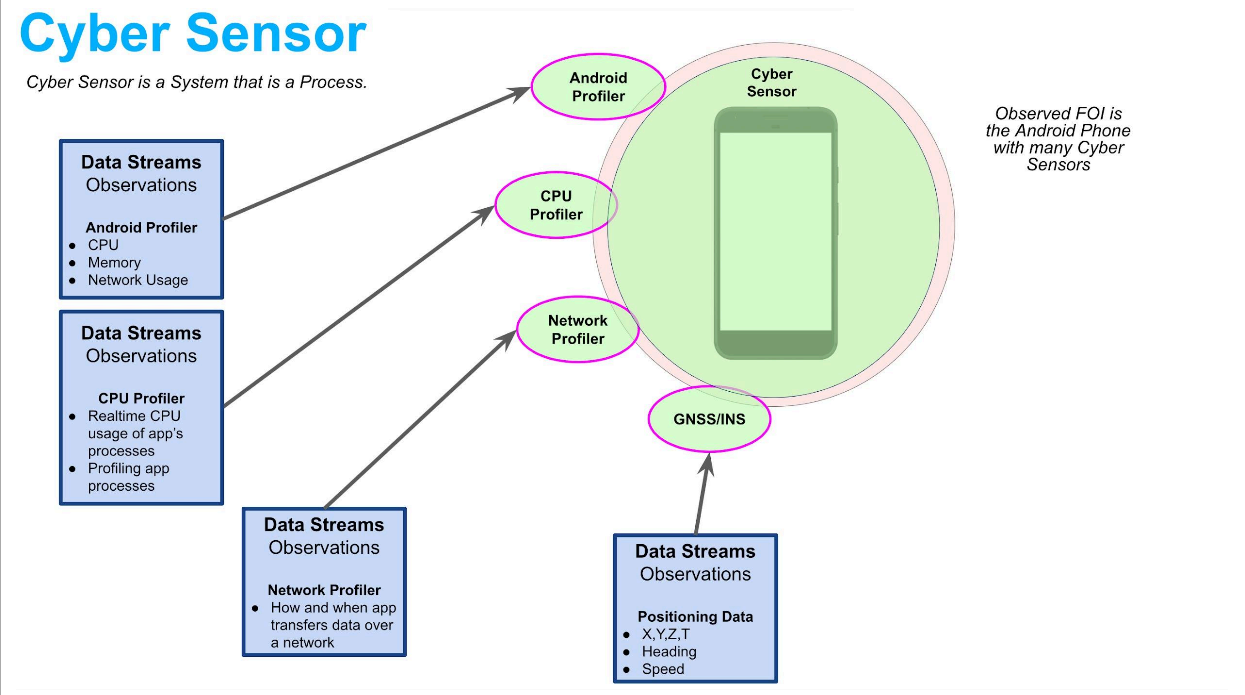
System	**Data Stream**	**Comments**
	Alert message sent to subscriber	(In the API, alerts are just Observations with a different meaning)

Control Streams/Commands:

System	Control Stream	Comments
	None	

7.1.21. Cyber Sensor

A Cyber Sensor is a System that is a Process that makes Observations about the state of a given device's software, data and network behaviors. The diagram and discussion below help convey how Cyber Sensors can be treated in the OGC API - Connected Systems Standard v1.0.


Cyber Sensor
Cyber Sensor is a System that is a Process.
Android Profiler
Cyber Sensor
Observed FOI is the Android Phone with many Cyber Sensors
CPU Profiler
Network Profiler
GNSS/INS
Data Streams Observations
Android Profiler
CPU
Memory
Network Usage
Data Streams Observations
CPU Profiler
Realtime CPU usage of app's processes
Profiling app processes
Data Streams Observations
Network Profiler
How and when app transfers data over a network
Data Streams Observations
Positioning Data
X,Y,Z,T
Heading
Speed

System: Top level System is a Process, with Sensor and Processing components
Platform: None
Sensors: Thermometer
Actuators: None
Processes: Process is a threshold crossing algorithm.
Features of Interest: Temperature sampling location

Data Streams/Observations: Alert message sent to subscriber. (In the API, alerts are just observations with a different meaning)
Control Streams/Commands: None.

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Alerts/Notification	System	
Alerts/Notification	Process	the top level System is the Platform in this case
Thermometer	Sensor	Subsystem mounted on the Platform
Threshold crossing algorithm	Process	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Temperature sampling location	Temperature Sensor	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

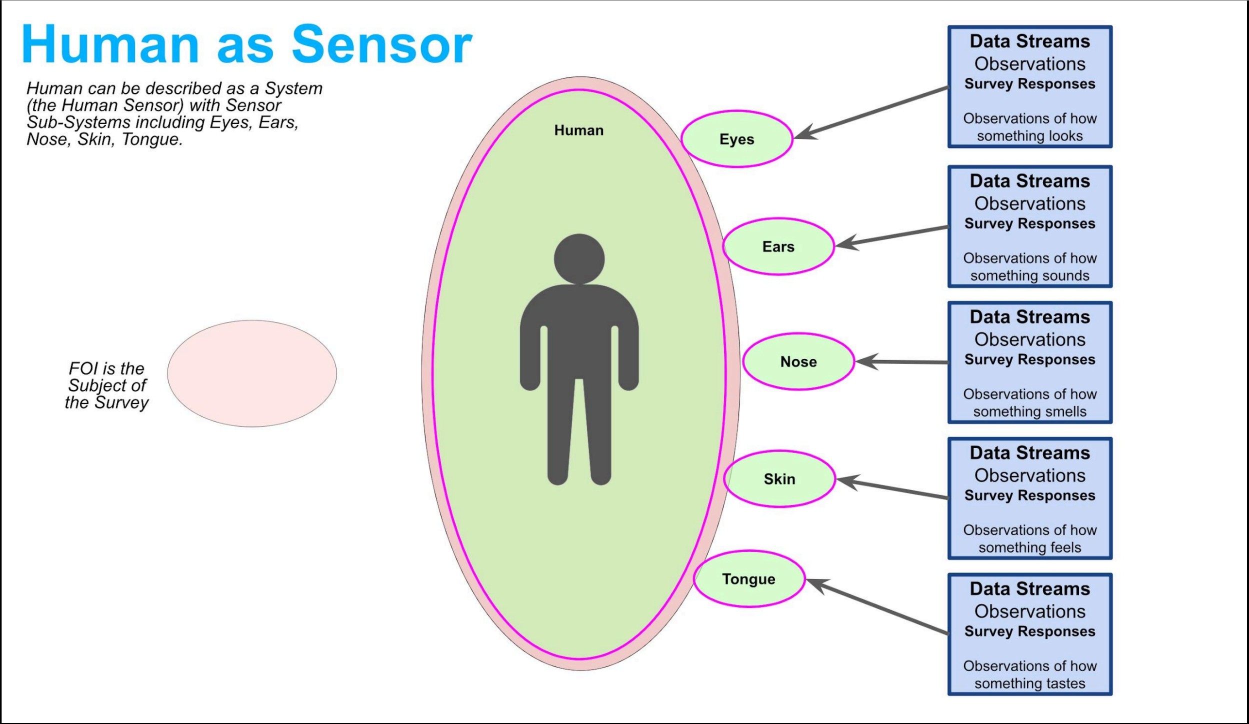
System	**Data Stream**	**Comments**
	Alert message sent to subscriber	(In the API, alerts are just observations with a different meaning)

Control Streams/Commands:

System	Control Stream	Comments
	None	

7.1.22. Human as Sensor

A Human is a System that is a Sensor capable of observing the world with eyes, ears, nose, skin, and tongue in order to make Observations of how a Feature of Interest looks, sounds, smells, feels, and tastes and input them into a survey mechanism. The diagram and discussion below help convey how Humans as Sensors can be treated in the OGC API - Connected Systems Standard v1.0.


Human as Sensor
Human can be described as a System (the Human Sensor) with Sensor Sub-Systems including Eyes, Ears, Nose, Skin, Tongue.
FOI is the Subject of the Survey
Human
Eyes
Ears
Nose
Skin
Tongue
Data Streams Observations Survey Responses Observations of how something looks
Data Streams Observations Survey Responses Observations of how something sounds
Data Streams Observations Survey Responses Observations of how something smells
Data Streams Observations Survey Responses Observations of how something feels
Data Streams Observations Survey Responses Observations of how something tastes

System: The top level System is the human Sensor
Platform: None
Sensors: The human filling up a survey based on observations from their eyes, ears, nose, skin, tongue.
Actuators: None
Processes: None
Features of Interest: Subject of the survey

Data Streams/Observations: Survey responses
Control Streams/Commands: None

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Human as Sensor	System	
Human	Sensor	the top level System is the Platform in this case
Eyes	Sensor	Subsystem is mounted on the Platform
Ears	Sensor	Subsystem is mounted on the Platform
Nose	Sensor	Subsystem is mounted on the Platform
Skin	Sensor	Subsystem is mounted on the Platform
Tongue	Sensor	Subsystem is mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Human (Sensor)	Human (eyes, ears, nose, skin, tongue)	
Subject of the survey	Human (eyes, ears, nose, skin, tongue)	

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

System	**Data Stream**	**Comments**

Human (eyes)	Survey responses	Observations of how something looks
Human (ears)	Survey responses	Observations of how something sounds
Human (nose)	Survey responses	Observations of how something smells
Human (skin)	Survey responses	Observations of how something feels
Human (tongue)	Survey responses	Observations of how something tastes

Control Streams/Commands:

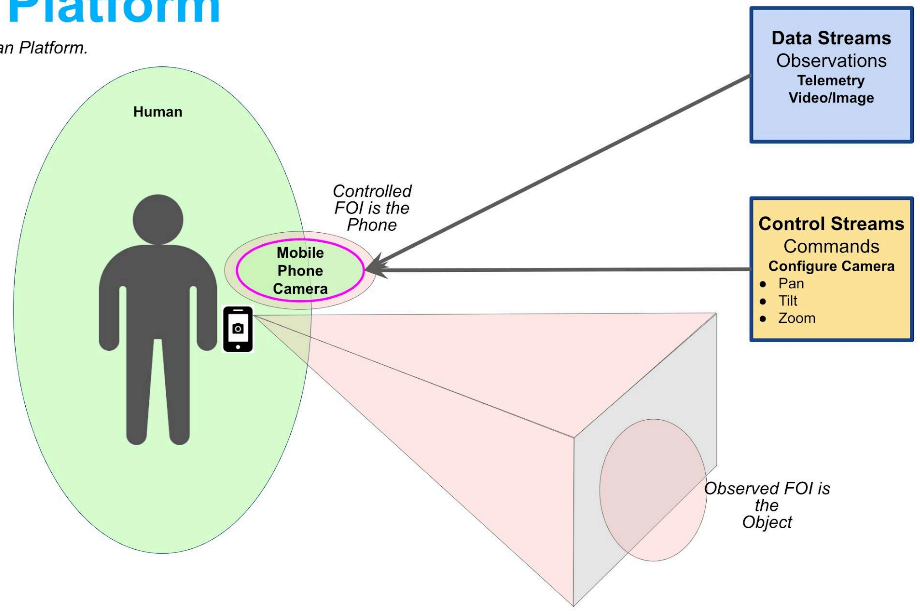
System	Control Stream	Comments
	None	

7.1.23. Human as Platform

A Human is a Platform for mounting/carrying (in this use case) a Sensor (mobile phone camera) that can be tasked to Pan, Tilt and Zoom. In this use case, the top level system is the Human Platform. The diagram and discussion below help convey how Humans as Platforms can be treated in the OGC API - Connected Systems Standard v1.0.

Human as Platform

The top level System is the Human Platform.



System: Top level System is the Human Platform
Platform: Human
Sensors: Mobile Phone Camera
Actuators: None
Processes: None
Features of Interest: Camera Frustum, Object
Data Streams/Observations: Mobile Phone Positioning Data, Video/Image
Control Streams/Commands: Command the Pan, Tilt and Zoom of the Camera, Configure Camera Video

Systems:

Name	**Type**	**Description (+ link to datasheet)**
Human (Platform)	System	
Human	Platform	the top level System is the Platform in this case
Mobile Phone Camera	Sensor	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to*	**System**	**Comments**

observe)		
Camera Frustum	Mobile Phone Camera	
Object	Mobile Phone Camera	Camera provides video/imagery of the object

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
Camera Frustum	Human hands	

Data Streams/Observations:

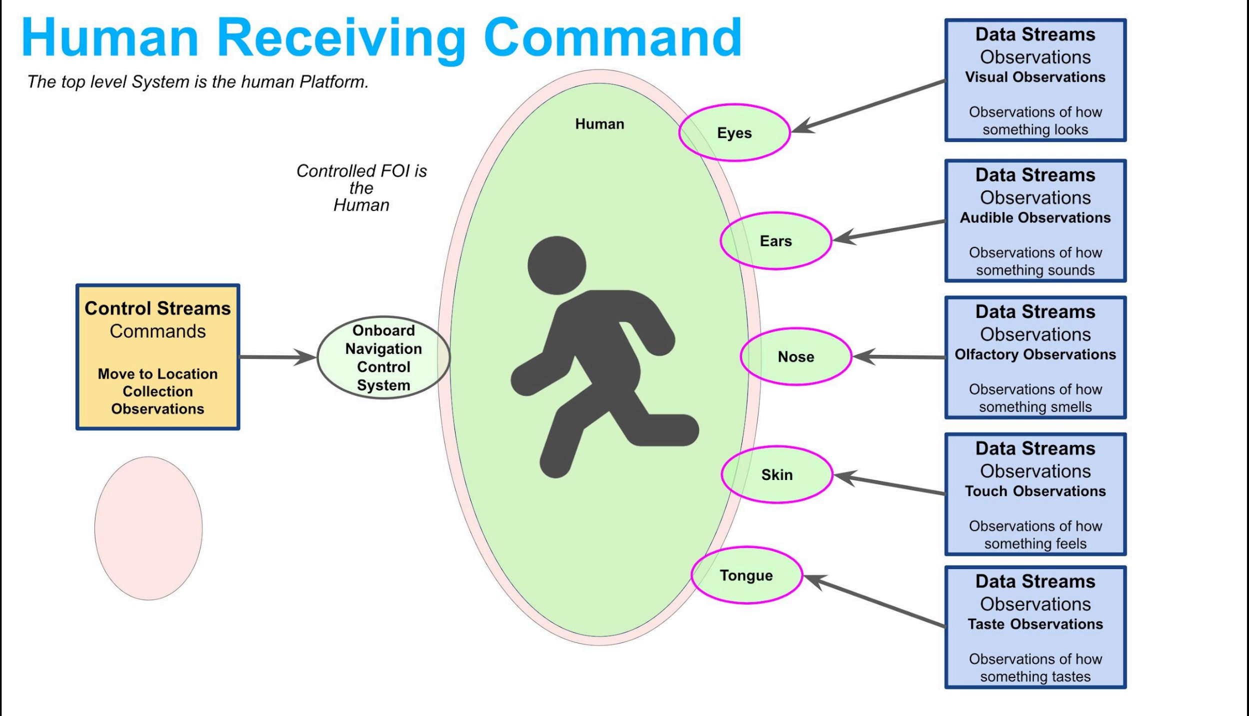
System	**Data Stream**	**Comments**
GNSS/INS	Mobile Phone Positioning Data	
Video Camera	Video/Image	

Control Streams/Commands:

System	**Control Stream**	**Comments**
Human hand	Command the Pan, Tilt and Zoom of the Camera	
Human hand	Configure Camera Video	

7.1.24. Human Receiving Command

A Human is a Platform that can be tasked to go somewhere at a time and collect Observations about a Feature of Interest with the Sensors inherent to the Human Platform/System - eyes, ears, nose, skin, and tongue - and/or to undertake some sort of action. The diagram and discussion below help convey how Humans (as Platforms) can receive Commands within the OGC API - Connected Systems Standard v1.0.


Human Receiving Command
The top level System is the human Platform.
Controlled FOI is the Human
Human
Control Streams
Commands
Move to Location Collection Observations
Onboard Navigation Control System
Eyes
Ears
Nose
Skin
Tongue
Data Streams
Observations
Visual Observations
Observations of how something looks
Data Streams
Observations
Audible Observations
Observations of how something sounds
Data Streams
Observations
Olfactory Observations
Observations of how something smells
Data Streams
Observations
Touch Observations
Observations of how something feels
Data Streams
Observations
Taste Observations
Observations of how something tastes

System: Top level System is the Human Platform
Platform: Human
Sensors: Eyes, Ears, Nose, Skin, Tongue
Actuators: Human legs, being told to go somewhere and sense/do something.
Processes: None
Features of Interest: Object
Data Streams/Observations: None
Control Streams/Commands: Location to move to received by the human

Systems:

Name	Type	Description (+ link to datasheet)
Human (Platform)	System	
Human	Platform	the top level System is the Platform in this case
Human l and hands, legs and arms	Actuator	Subsystem mounted on the Platform

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Object		

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
Human		Task the human to go somewhere at a moment of time and collect data or do something.

Data Streams/Observations:

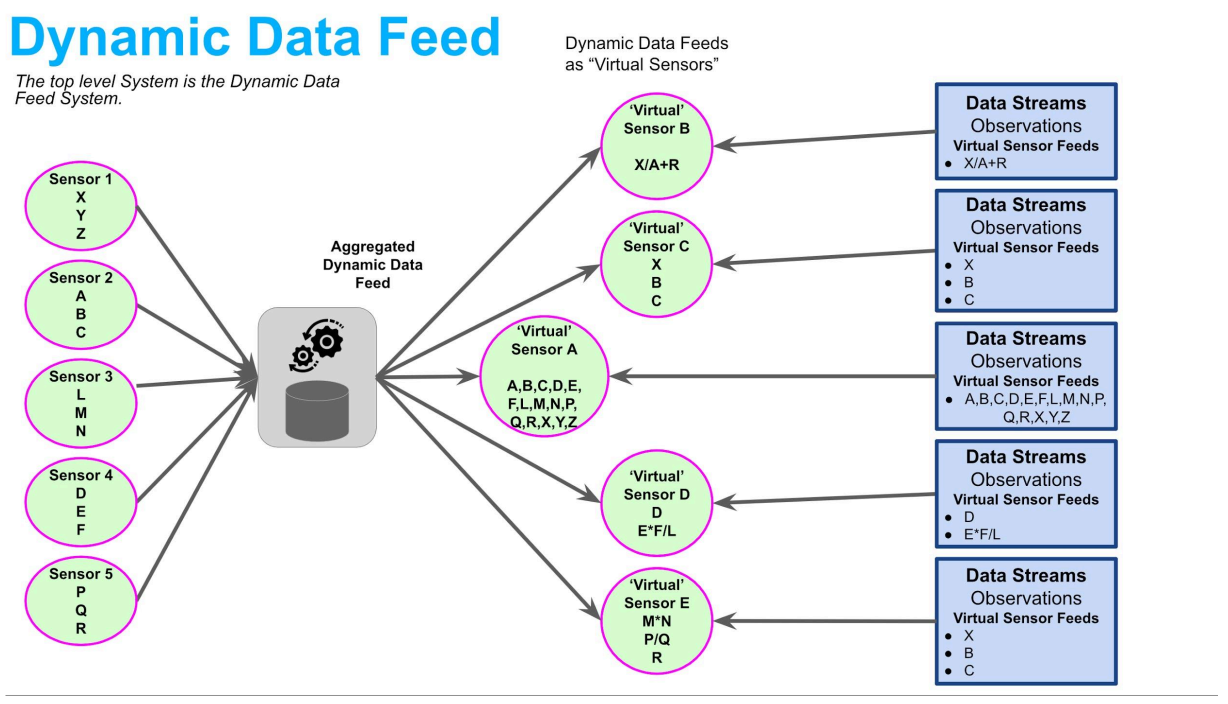
System	**Data Stream**	**Comments**
Eyes	Visual observations	Observations of how something looks
Ears	Audible observations	Observations of how something sounds
Nose	Olfactory observations	Observations of how something smells
Skin	Temperature, texture, pressure, etc. observations	Observations of how something feels
Tongue	Taste observations	Observations of how something tastes

Control Streams/Commands:

System	**Control Stream**	**Comments**
	Command to human to move to a location and collect observations.	

7.1.25. Dynamic Data Feed

A Dynamic Data Feed is a System that is an aggregation of multiple underlying Sensor Data Streams that can be either aggregated into a single ‘Virtual’ Sensor, or recombined in different ways into n- Virtual Sensors. The diagram and discussion below help convey how Dynamic Data Feeds can be treated in the OGC API - Connected Systems Standard v1.0.


Dynamic Data Feed
The top level System is the Dynamic Data Feed System.
Dynamic Data Feeds as “Virtual Sensors”
Sensor 1 X Y Z
Sensor 2 A B C
Sensor 3 L M N
Sensor 4 D E F
Sensor 5 P Q R
Aggregated Dynamic Data Feed
‘Virtual’ Sensor A A,B,C,D,E, F,L,M,N,P, Q,R,X,Y,Z
‘Virtual’ Sensor B X/A+R
‘Virtual’ Sensor C X B C
‘Virtual’ Sensor D D E*F/L
‘Virtual’ Sensor E M*N P/Q R
Data Streams Observations Virtual Sensor Feeds • X/A+R
Data Streams Observations Virtual Sensor Feeds • X • B • C
Data Streams Observations Virtual Sensor Feeds • A,B,C,D,E,F,L,M,N,P, Q,R,X,Y,Z
Data Streams Observations Virtual Sensor Feeds • D • E*F/L
Data Streams Observations Virtual Sensor Feeds • X • B • C

System: Top level System is the Dynamic Data Feed
Platform: System
Sensors: Sensors 1-5, ‘Virtual’ Sensors A-E
Actuators: None
Processes: None
Features of Interest: Object
Data Streams/Observations: None
Control Streams/Commands: None

Systems:

Name	Type	Description (+ link to datasheet)
Dynamic Data	System	

Feed (Platform)		
Dynamic Data Feed	Platform	the top level System is the Platform in this case
Underlying Sensors	Sensor	
Virtual Sensors	Sensor	

Features of Interest:

Observed FOI *(the thing you want to observe)*	**System**	**Comments**
Object	Virtual Sensor	Each Virtual Sensor will have Observed FOI.

Controlled FOI *(the thing you want to control)*	**System**	**Comments**
	None	

Data Streams/Observations:

System	**Data Stream**	**Comments**
Virtual Sensor	Yes	Each Virtual Sensor will have Data Streams/Observations.

Control Streams/Commands:

System	**Control Stream**	**Comments**
	Command to human to move to a location and collect observations.	

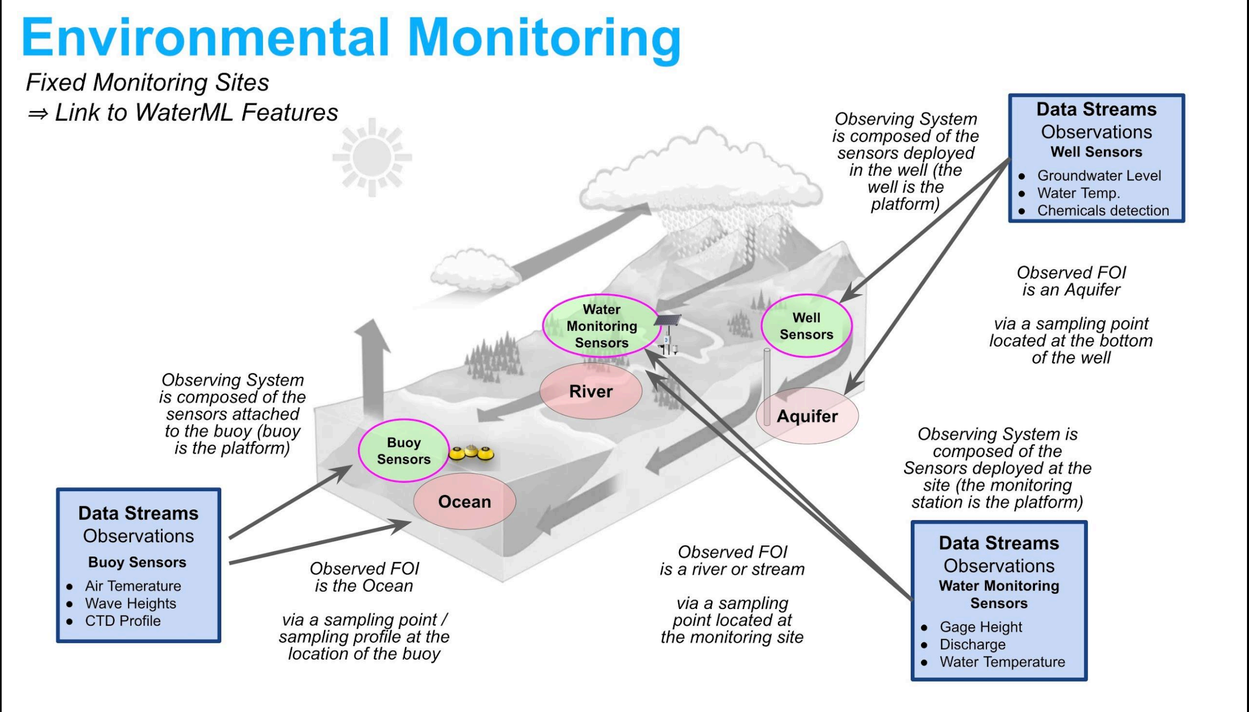
7.2. Domain use cases

This section provides concrete domain use cases of how Systems, Platforms, Sensors, Processes, Actuators, Features of Interest, Data Streams and their Observations, and Control Streams and their Commands work together when integrating different kinds of systems via the OGC API - Connected Systems Standard v1.0 within and across a particular domain. These include:

1) Environmental Monitoring
2) Logistics
3) Energy and Utilities
4) Facility/Installation/Campus Security
5) Smart Cities
6) Industrial Monitoring and Control (IoT/SCADA)
7) Maritime Domain Awareness
8) Joint All Domain Command and Control
9) Smart Buildings
10) Aviation

7.2.1. Environmental Monitoring

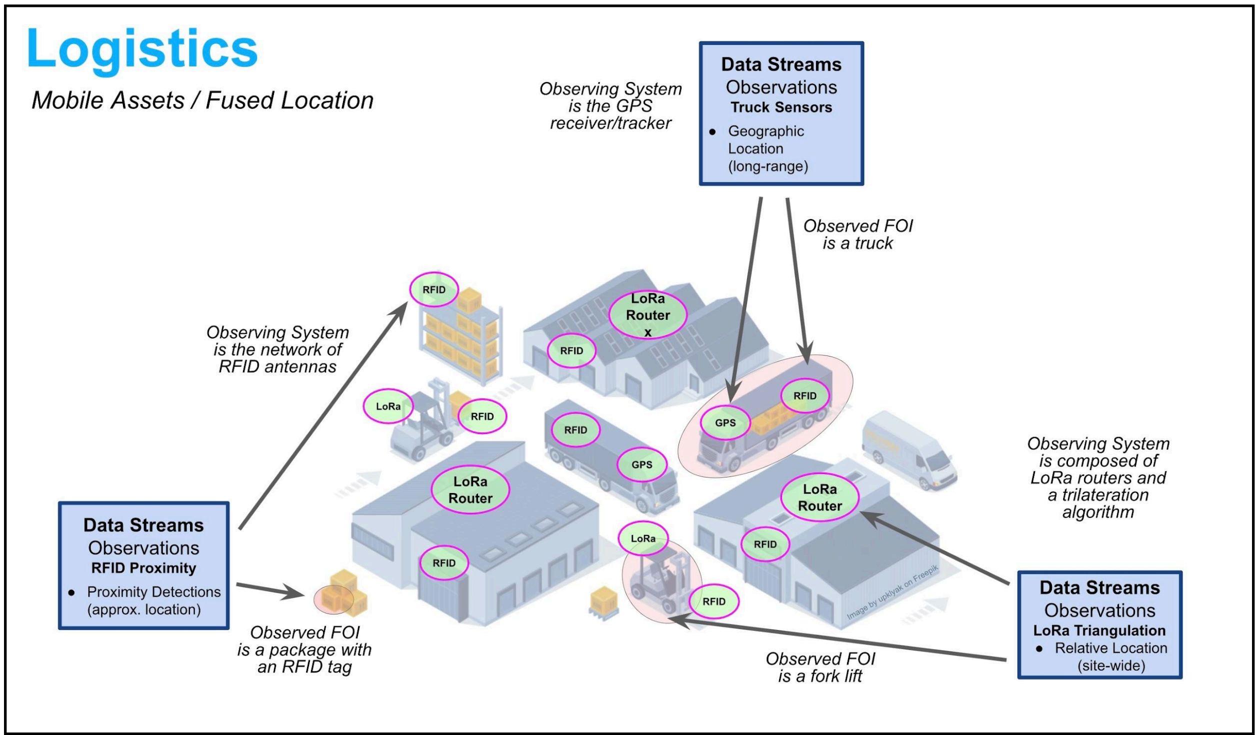
Monitoring environmental change requires the integration of many sensing modalities within a common 4D framework. An OGC standards-based interoperability architecture for environmental monitoring enables the integration of all kinds of Systems. The OGC API - Connected Systems Standard v1.0 offers architectural opportunities to enable the rapid collection, fusion, and customization of integrated sensed Observations from every source within a common 4D framework.


Environmental Monitoring
Fixed Monitoring Sites
⇒ Link to WaterML Features
Observing System is composed of the sensors deployed in the well (the well is the platform)
Data Streams
Observations
Well Sensors
• Groundwater Level
• Water Temp.
• Chemicals detection
Observed FOI is an Aquifer
via a sampling point located at the bottom of the well
Water Monitoring Sensors
Well Sensors
River
Aquifer
Observing System is composed of the sensors attached to the buoy (buoy is the platform)
Buoy Sensors
Ocean
Observing System is composed of the Sensors deployed at the site (the monitoring station is the platform)
Data Streams
Observations
Buoy Sensors
• Air Temerature
• Wave Heights
• CTD Profile
Observed FOI is the Ocean
via a sampling point / sampling profile at the location of the buoy
Observed FOI is a river or stream
via a sampling point located at the monitoring site
Data Streams
Observations
Water Monitoring Sensors
• Gage Height
• Discharge
• Water Temperature

System: Each Sensor is a System
Platform: None
Sensors: Buoy Sensors, Water Monitoring Sensors, Well Sensors
Processes: None
Actuators: None
Features of Interest: Ocean, River, Aquifer
Data Streams/Observations: Air Temperature, Wave Heights, CTD Profile (Buoy Sensor), Gauge Height, Discharge, Water Temperature (Water Monitoring Sensors), Groundwater Level, Water Temperature, Chemicals detection (Well Sensors)
Control Streams/Commands: None

7.2.2. Logistics

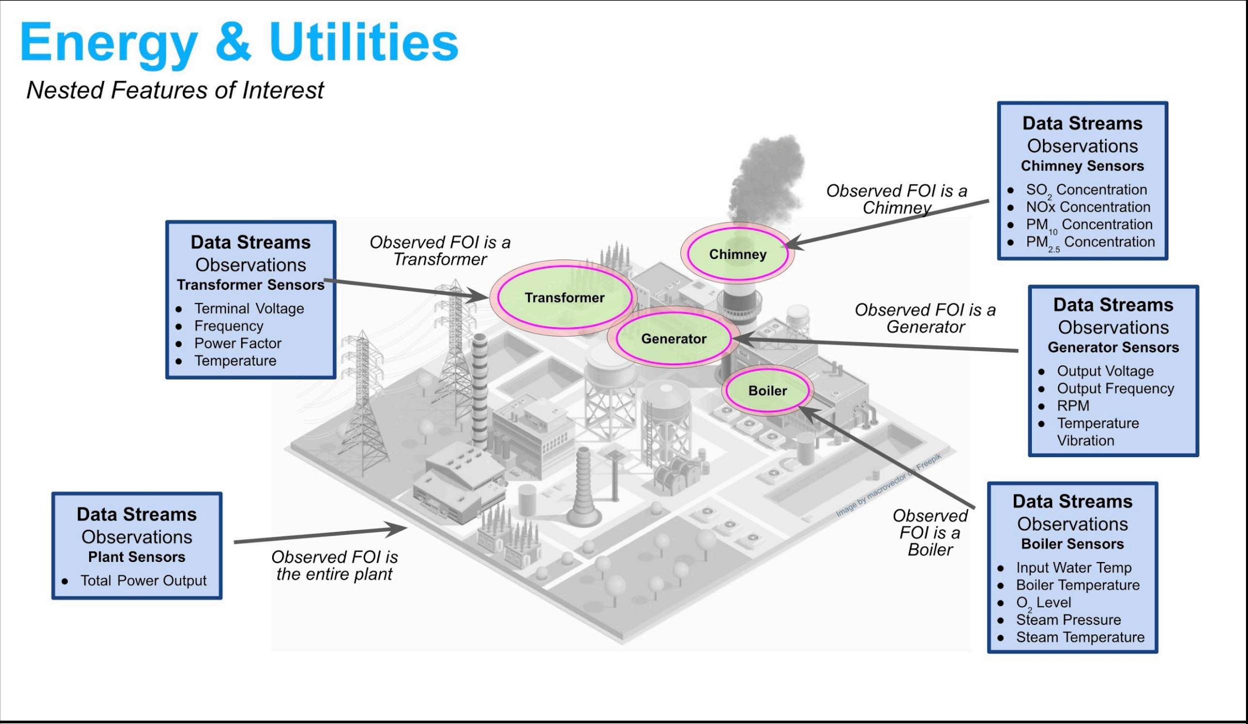
Managing logistics across complex supply chains requires detailed tracking of goods/freight/cargo at a very granular level as these items move from one origin facility to a destination, often through many intermediate locations, on one or more Platforms, and even within intermediate Platforms such as shipping containers. The OGC API - Connected System Standard v1.0 offers architectural opportunities to enable the rapid collection, fusion, and customization of integrated sensed Observations from every source within a common 4D framework.


Logistics
Mobile Assets / Fused Location
Observing System is the GPS receiver/tracker
Data Streams Observations Truck Sensors
• Geographic Location (long-range)
Observed FOI is a truck
Observing System is the network of RFID antennas
RFID
LoRa Router x
LoRa
GPS
LoRa Router
Observing System is composed of LoRa routers and a trilateration algorithm
Data Streams Observations RFID Proximity
• Proximity Detections (approx. location)
Observed FOI is a package with an RFID tag
Observed FOI is a fork lift
Data Streams Observations LoRa Triangulation
• Relative Location (site-wide)
Image by upklyak on Freepik

System: Each Sensor is a System
Platform: None
Sensors: GPS, RFID, LoRA
Processes: None
Actuators: None
Features of Interest: Truck, Fork Lift, Package
Data Streams/Observations: Geographic Location (Long-Range, Truck), Relative Location (Site-Wide, Fork Lift), Proximity Detections (Approximate Location, Package)
Control Streams/Commands: None

7.2.3 Energy & Utilities

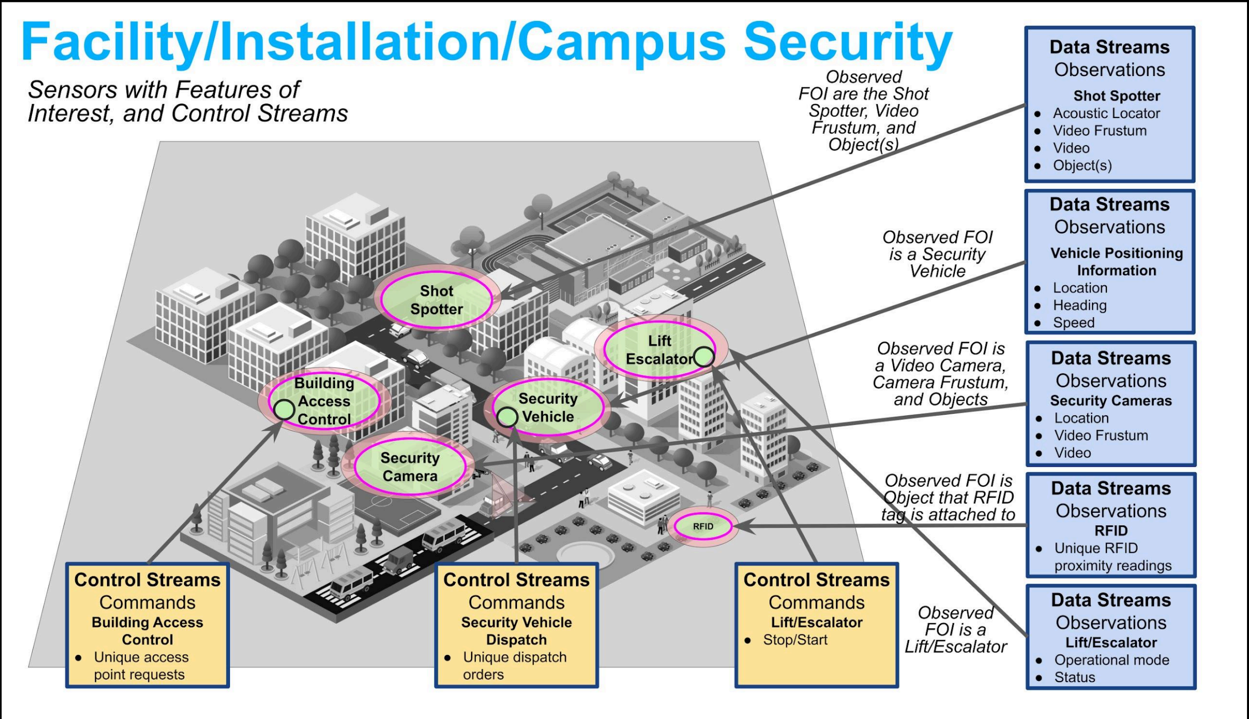
The generation, distribution and use of energy can be a geographically complex endeavor, with different patterns for those energy utilities requiring fuel sources. The OGC API - Connected System Standard v1.0 offers architectural opportunities to enable the rapid collection, fusion, and customization of integrated sensed Observations from every source within a common 4D framework.



System: Each Sensor is a System
Platform: None
Sensors: Boiler Sensors, Generator Sensors, Chimney Sensors, Transformer Sensors, Plant Sensors
Processes: None
Actuators: None
Features of Interest: Boiler, Generator, Chimney, Transformer, Plant
Data Streams/Observations: Input Water Temp, Boiler Temperature, O_2 Level, Steam Pressure, Steam Temperature (Boiler); Output Voltage, Output Frequency, RPM, Temperature, Vibration (Generator); SO_2, NOx, PM_{10},$PM_{2.5}$ Concentration (Chimney); Terminal Voltage, Frequency, Power Factor, Temperature (Transformer); Total Power Output (Plant)
Control Streams/Commands: None

7.2.4. Facility/Installation/Campus Security

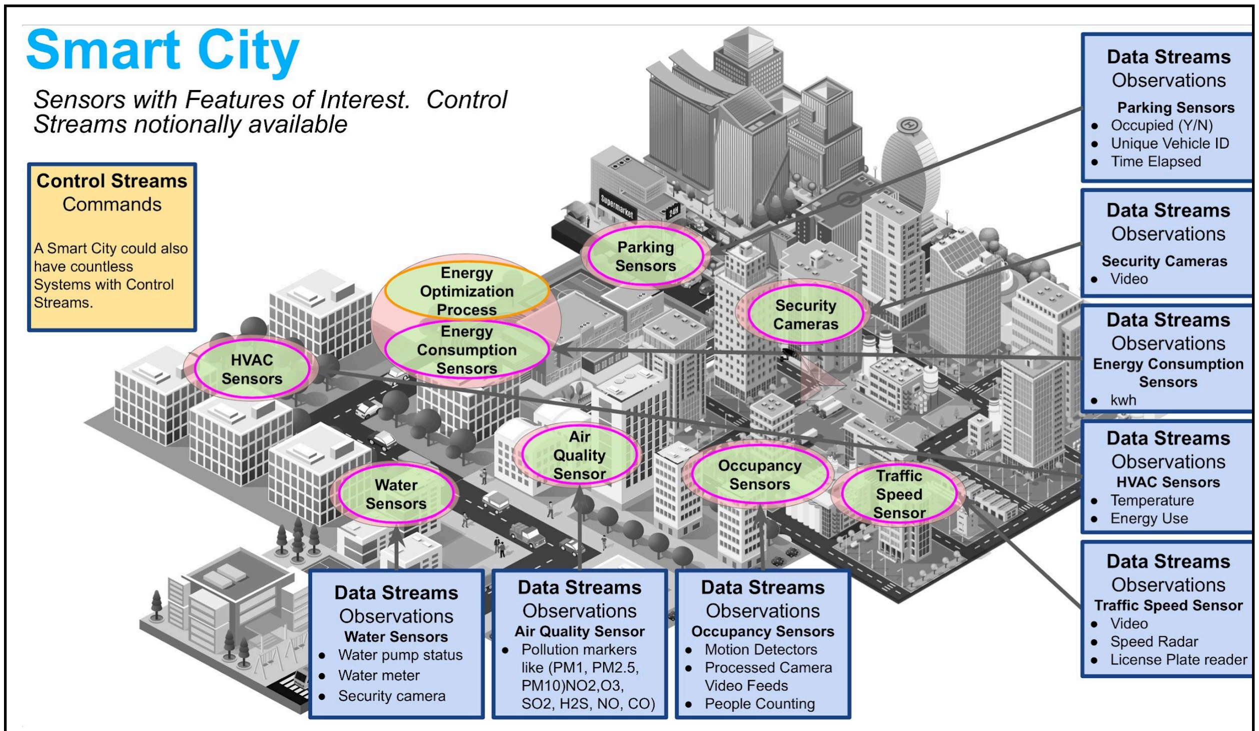
Securing facilities, installations, and campuses requires the integration and dynamic tasking of a variety of different kinds of sensors, control systems, and response resources. The OGC API - Connected System Standard v1.0 offers architectural opportunities to enable the rapid collection, fusion, and customization of integrated sensed Observations from every source within a common 4D framework.


Facility/Installation/Campus Security
Sensors with Features of Interest, and Control Streams
Observed FOI are the Shot Spotter, Video Frustum, and Object(s)
Data Streams Observations Shot Spotter
Acoustic Locator
Video Frustum
Video
Object(s)
Observed FOI is a Security Vehicle
Data Streams Observations Vehicle Positioning Information
Location
Heading
Speed
Observed FOI is a Video Camera, Camera Frustum, and Objects
Data Streams Observations Security Cameras
Location
Video Frustum
Video
Observed FOI is Object that RFID tag is attached to
Data Streams Observations RFID
Unique RFID proximity readings
Observed FOI is a Lift/Escalator
Data Streams Observations Lift/Escalator
Operational mode
Status
Shot Spotter
Lift Escalator
Building Access Control
Security Vehicle
Security Camera
RFID
Control Streams Commands Building Access Control
Unique access point requests
Control Streams Commands Security Vehicle Dispatch
Unique dispatch orders
Control Streams Commands Lift/Escalator
Stop/Start

System: Each Sensor is a System
Platform: None
Sensors: Shot Spotter, Vehicle Positioning Information, Security Cameras, RFID
Processes: None
Actuators: Building Access Control, Security Vehicle Dispatch, Lift, Escalator
Features of Interest: Shot Spotter, Shot Spotter Video Frustum, Object(s), Security Vehicle, Video Camera, Camera Frustum, Object(s), Objects tagged with RFID.
Data Streams/Observations: Acoustic Locator, Video Frustum, Video, Object(s) (Shot Spotter); Location, Heading, Speed (Vehicle Positioning Information), Location, Video Frustum, Video-RSTP (Security Cameras), Unique RFID proximity readings (RFID), Operational mode and status (Lift, Escalator)
Control Streams/Commands: Unique access point requests (Building Access Control), Unique dispatch orders (Security Vehicle Dispatch), Mode change requests (Lift)

7.2.5 Smart Cities

The management of smart cities requires the integration and dynamic tasking of a variety of different kinds of sensors, control systems, and response resources. The OGC API - Connected System Standard v1.0 offers architectural opportunities to enable the rapid collection, fusion, and customization of integrated sensed Observations from every source within a common 4D framework.



System: Each Sensor is a System
Platform: None
Sensors: Parking Sensors, Security Cameras, Energy Consumption Sensors, HVAC Sensors, Traffic Speed Sensor, Air Quality Sensor, Occupancy Sensors, Water Sensors
Processes: Energy Optimisation Process
Actuators: None
Features of Interest: Parking Sensors, Security Cameras, Energy Consumption Sensors, HVAC Sensors, Traffic Speed Sensor, Air Quality Sensor, Occupancy Sensors, Water Sensors, Security Camera
Data Streams/Observations: Occupied (Y/N), Unique Vehicle ID, Time Elapsed (Parking Sensors); Video (Security Cameras); kWh (Energy Consumption Sensors); Temperature, Energy Use (HVAC Sensors); Video, Speed Radar, License Plate Reader (Traffic Speed Sensor); Pollution Markers (Air Quality Sensor), Video-RSTP (Security camera), Motion Detector, Processed Camera Video Feeds, People Counting (Occupancy Sensors), Water Pump Status, Water Meter, Security Camera (Water Sensors)
Control Streams/Commands: Could have, but not assumed in this use case.

7.2.6. Industrial Monitoring and Control (IoT/SCADA)

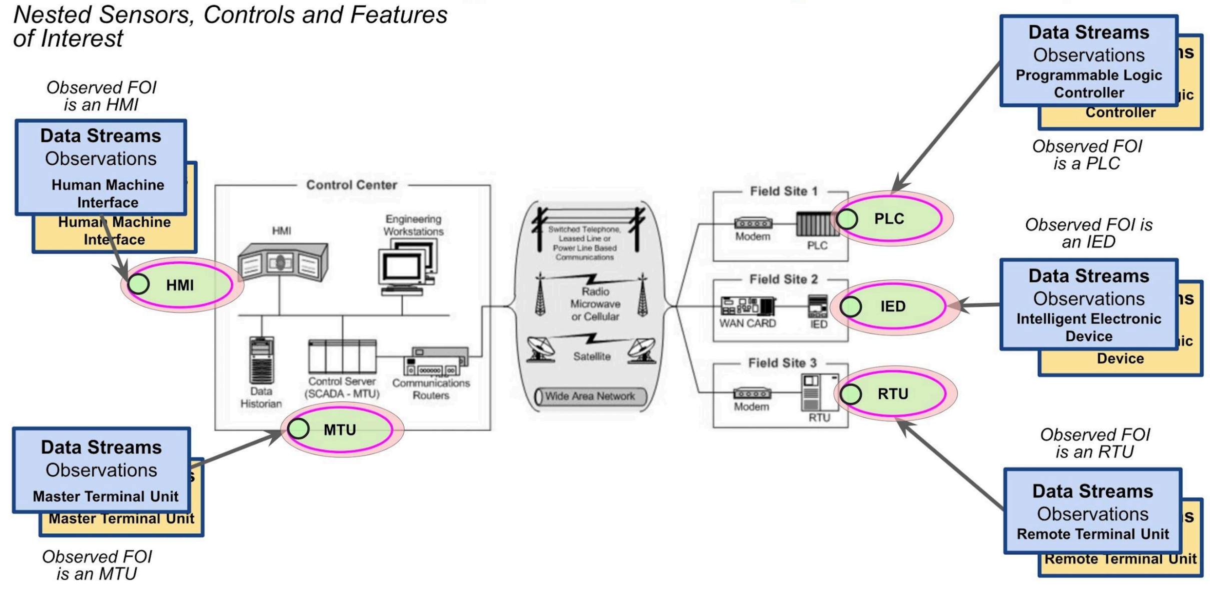
Industrial facilities, infrastructure and processes require active monitoring and control. Historically, this required Supervisory Control and Data Acquisition (SCADA) Systems, while now the conversation centers more on “industrial IoT”. Both are simply constellations of Sensors, Processes and Actuators arrayed across a complex industrial infrastructure in order to observe, make sense of, and take actions that drive efficiency, error mitigation, safety, profitability, and overall effectiveness.

Supervisory Control and Data Acquisition (SCADA) Systems are used for controlling, monitoring, and analyzing industrial devices and processes. The system consists of both software and hardware components and enables remote and on-site gathering of data from the industrial equipment. The connecting links in the SCADA architecture, which connect to equipment (also called field devices) are commonly termed the Programmable Logic Controllers (PLCs), Intelligent Electronic Devices (IED), Remote Terminal Units (RTUs), Master Terminal Units (MTU) which in turn connect to Human Machine Interfaces (HMIs). SCADA Systems are using in Manufacturing, Water Management, Oil and Gas, Transportation, Renewable Energy, Power distributions and control.

The OGC API - Connected System Standard v1.0 offers architectural opportunities to enable rapid collection, fusion, and customization of integrated SCADA Observations from every source within a common 4D framework.

Industrial Monitoring and Control (IoT/SCADA)

Nested Sensors, Controls and Features of Interest



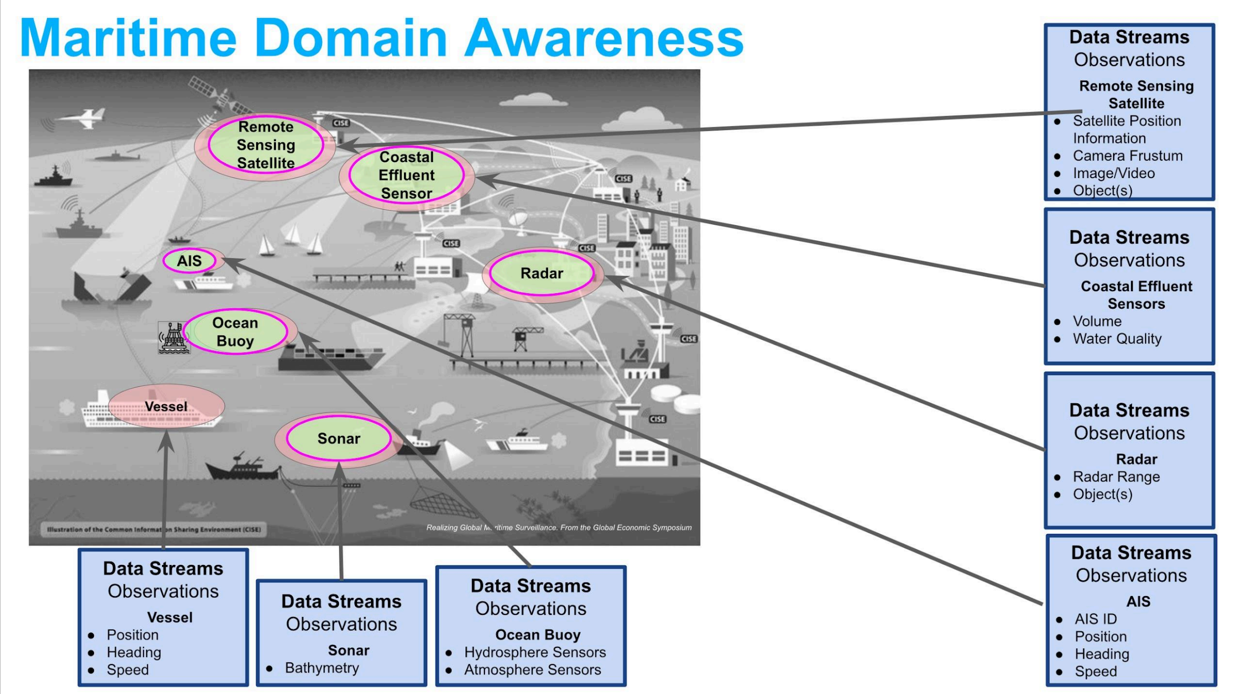
SCADA System General Layout, NIST (Stouffer, Falco, and Scarfone, 2011
https://www.researchgate.net/figure/SCADA-System-General-Layout-NIST-Stouffer-Falco-and-Scarfone-2011_fig1_31666648

System: System comprises a variety of Subsystems that are both Sensors, Actuators, and Features of Interest.
Platform: None
Sensors: HMI, MTU, PLC, IED, RTU
Processes: None
Actuators: HMI, MTU, PLC, IED, RTU
Features of Interest: HMI, MTU, PLC, IED, RTU

Data Streams/Observations: HMI, MTU, PLC, IED, RTU
Control Streams/Commands: HMI, MTU, PLC, IED, RTU

7.2.7. Maritime Domain Awareness

Maritime domain awareness is important for commercial maritime operations, coast guard and law enforcement operations, and national security operations. Maritime domain awareness is achieved through the integration of space-based, airborne, mobile/marine, in situ and terrestrial/marine remote sensors of a wide variety of phenomenologies. With the rising prevalence of USV and UUV, as well as UAS, within the maritime domain, the tasking of such Platforms must also be taken into account. The OGC API - Connected System Standard v1.0 offers architectural opportunities to enable the rapid collection, fusion, and customization of integrated sensed observations from every Maritime Domain Awareness source within a common 4D framework.



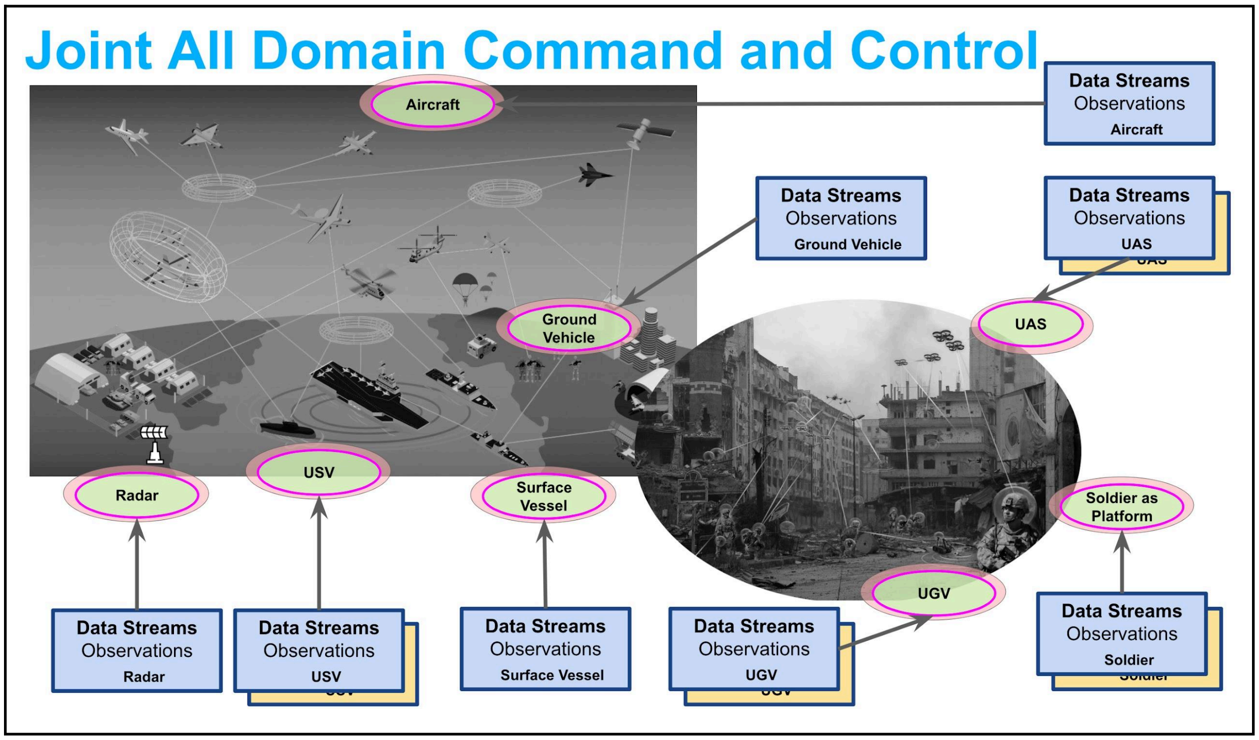
Maritime Domain Awareness
Remote Sensing Satellite
Coastal Effluent Sensor
AIS
Radar
Ocean Buoy
Vessel
Sonar
CISE
Illustration of the Common Information Sharing Environment (CISE)
Realizing Global Maritime Surveillance. From the Global Economic Symposium
Data Streams Observations
Remote Sensing Satellite
Satellite Position Information
Camera Frustum
Image/Video
Object(s)
Data Streams Observations
Coastal Effluent Sensors
Volume
Water Quality
Data Streams Observations
Radar
Radar Range
Object(s)
Data Streams Observations
AIS
AIS ID
Position
Heading
Speed
Data Streams Observations
Vessel
Position
Heading
Speed
Data Streams Observations
Sonar
Bathymetry
Data Streams Observations
Ocean Buoy
Hydrosphere Sensors
Atmosphere Sensors

System: Each Sensor is a System
Platform: None
Sensors: Remote Sensing Satellite, Coastal Effluent Sensor, Radar, AIS, Sonar
Processes: None
Actuators: None
Features of Interest: Remote Sensing Satellite, Coastal Effluent Sensor, Radar, AIS, Sonar, Vessel

Data Streams/Observations: Satellite Position Information, Camera Frustum, Image/Video, Object(s) (Remote Sensing Satellite); Volume, Water Quality (Coastral Effluent Sensors); Radar Range, Object(s) (Radar); AIS ID, Position, Heading, Speed (AIS); Hydrosphere Sensors, Atmosphere Sensors (Ocean Boy); Bathymetry (Sonar); Position, Heading, Speed (Vessel) **Control Streams/Commands:** None

7.2.8. Joint All Domain Command and Control

JADC2 demands an architecture that can sense and simultaneously integrate information from and within all domains to enable the Joint Force Commander to achieve information and decision advantage. "Sense and integrate" is the ability to discover, collect, correlate, aggregate, process, and exploit data from all domains and sources (friendly, adversary, and neutral), and share the information as the basis for understanding and decision-making. OGC standards-based interoperability architecture enables the integration of sensors, things, robots, drones, satellites, control systems devices and Platforms across space, air, land, sea, cyber, and electro-magnetic spectrum - observing the world across all phenomenologies (e.g., EO, IR, MSI, HSI, LiDAR, Radar, SAR, GMTI SAR, Sonar, Acoustic, RF, CBRNE, health, cyber, etc.). OGC API - Connected Systems Standard v1.0 offers architectural opportunities to enable the rapid collection, fusion, and customization of integrated sensed observations from every source within a common 4D framework.


Joint All Domain Command and Control
Aircraft
Data Streams Observations Aircraft
Data Streams Observations Ground Vehicle
Data Streams Observations UAS
Ground Vehicle
UAS
Radar
USV
Surface Vessel
Soldier as Platform
UGV
Data Streams Observations Radar
Data Streams Observations USV
Data Streams Observations Surface Vessel
Data Streams Observations UGV
Data Streams Observations Soldier

System: Each Platform is a System.
Platform: Aircraft, Surface Vessel, Ground Vehicles, UAS, UGV, USV, Soldier, Radar
Sensors: Each Platform has many Sensors.
Processes: Each Platform has many Processes.
Actuators: Some Platforms have Actuators for Tasking.
Features of Interest: Aircraft, Surface Vessel, Ground Vehicles, UAS, UGV, USV, Soldier, Radar
Data Streams/Observations: Many
Control Streams/Commands: Many

8. Other SDOs

Beyond the Open Geospatial Consortium, maintaining the OGC API - Connected Systems Standard v1.0 will require actively engaging with the specifications, processes, and leadership of other standards development organizations (SDO). This will include:

IETF®	IETF	HTTP, TCP/IP, UDP, RTP, RTSP, SSL
W3C® World Wide Web consortium	World Wide Web Consortium	XML, WebSockets
OASIS OPEN	OASIS	MQTT AQMP
IEEE®	IEEE	HLA/DIS, etc.
ISO International Organization for Standardization IEC® International Electrotechnical Commission	ISO/IEC Moving Picture Experts Group	MPEG-4
	ISO/IEC JTC 1/SC 22, ICS 35.060	JSON
	ISO/IEC 19464	AMQP
MISB Motion Imagery Standards Board	MISB	H.264/MISB
Defence Geospatial Information Working Group DGIWG	DGIWG	STANAG 4609

SISO Simulation Interoperability Standards Organization	SISO	HLA/DIS (via IEEE)
THE Open GROUP	OpenGroup	SOSA
OMG Object Management Group	Object Management Group	DDC
KHRONOS GROUP	Khronos	glTF, Collada, WebGL
ROS Robot Operating System	ROS	Note: ROS is not a traditional ISO, but a community developing standard libraries, interfaces, and encodings for robotics.
MAVLINK MICRO AIR VEHICLE COMMUNICATION PROTOCOL	MAVLink	Note: MAVLink is not a traditional ISO, but a community developing standard libraries, interfaces, and encodings for UxS.
ARDUPILOT	ArduPilot	Note: ArduPilot is not a traditional ISO, but a community developing standard libraries, interfaces, and encodings for autopilot.
THE APACHE SOFTWARE FOUNDATION	Apache	Note: Apache is not a traditional ISO, but an open source software foundation which supports specifications like Apache Avro™.

9. Conclusion

The OGC API - Connected Systems Standard v1.0 provides the foundation for connecting all Systems in, on, and around our planet (and potentially other celestial bodies) within a common 4D framework for discovery, access, process, reasoning, visualization, tasking, and action. As more and varied Systems and technical communities come into existence, this standard will need to continue to evolve to ensure that all such Systems can interoperate within a common 4D framework with spatio-temporal precision precision and accuracy. After all, everything on Earth, by definition, exists in space and time, and as such all our Systems need to interoperate in this manner. Going forward all technical communities, user communities, and policy communities are invited to join and participate in the OGC Connected Systems Specification Working Group (SWG) that will govern the evolution of the OGC API - Connected Systems Standard v1.0.