SAPIENT Interface Control Document

DSTL/PUB145591
01-Feb-2023

© Crown-owned copyright (2015-2023), Dstl. This material is licensed under the terms of the Open Government Licence version 3 except where otherwise stated. To view this licence, visit http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3 or write to the Information Policy Team, The National Archives, Kew, London TW9 4DU, or email: psi@nationalarchives.gsi.gov.uk. Any contributions or requested ammendments to this document must be provided to Dstl under the Creative Commons Zero (CC-0) licence.

.

Administration Page

Customer Information	
Project title	SAPIENT
Customer Organisation	Dstl
Customer contact	Paul Thomas (SAPIENT@dstl.gov.uk, SAPIENT Interface Management Panel, Chairman)
	Stuart Colley (SAPIENT@dstl.gov.uk, SAPIENT Interface Management Panel, Secretary)

Principal authors	
Dr G.F. Marshall	QinetiQ Ltd, Malvern Technology Centre, St Andrews Road, Malvern, WR14 3PS
D.A.A. Faulkner	QinetiQ Ltd, Malvern Technology Centre, St Andrews Road, Malvern, WR14 3PS
T. Mann	QinetiQ Ltd, Malvern Technology Centre, St Andrews Road, Malvern, WR14 3PS

Release Authority	
Name	Paul Thomas
Post	Chairman, SAPIENT Interface Management Panel (SIMP)
Date of issue	See Record of Changes

Record of changes		
Issue	Date	Detail of Changes
1.0	10th Jan 2014	First issue
1.1	14th Jan 2014	For distribution to SAPIENT project
2.0	31st March 2014	Second issue, providing further clarification
3.0	7th November 2014	Third Issue for Review
3.1	18th December 2014	Revised following Review
4.0	22nd October 2015	Minor revisions following September 2015 Demonstration
4.1	27th November 2015	Updated for wider release
5.0-DRAFT	29th August 2019	Draft Rewrite to simplify and clarify ICD and move Taxonomy to Appendix
5.0	11th May 2020	Version 5 approved by SAPIENT Interface Management Panel (SIMP)
6.0	17th December 2021	MOD C-sUAS CDC changes incorporated
7.0	1st January 2023	Protobuf adopted as message format

List of Contents

1 Introduction

1.1 The SAPIENT Project

Sensing for Asset Protection with Integrated Electronic Networked Technology (SAPIENT) is a concept that combines modular autonomous sensing with fusion and sensor management. Since 2013 the SAPIENT project has developed the concept, standards and demonstrator systems that embody this concept. The key principles of SAPIENT are:

- Reduction in operator workload during monitoring activities;
- Improved autonomy for decision making (automatic identification of threat activities to reduce false alarm rates, allowing for autonomous context driven decision making);
- Improved autonomy for sensor management;
- Lower bandwidth requirements for network traffic;
- Sensor modularity – sensors of different types use the same interface for communication.

The SAPIENT project has investigated the following use cases:

- Protection of land-based high-value assets with defined borders;
- Situational Awareness over large areas;
- Counter Unmanned Aerial Systems (C-UAS);
- Situation Awareness in contested urban environments.

The SAPIENT concept considers a system of systems approach, consisting of multiple Autonomous Sensor Modules (ASM) that are all connected to a single High Level Decision Making Module (HLDMM).

1.2 Purpose of the document

This document describes the interface specification for SAPIENT components (middleware, sensor or effector nodes, and decision making modules).

This version of the SAPIENT ICD superceeds previous versions and is published on the official UK government (.gov.uk) website where a link to the test harness and middleware software can also be found.

1.3 Scope of the document

This document covers the interfaces between sensor or effector Nodes (historically, sensors have been refered to as Autonomous Sensor Modules (ASMs)), the Decision Making Modules (DMM) and the SAPIENT Middleware.

The SAPIENT concept is intended to support a range of different sensing modalities, a significant range of levels of object detection, localisation and classification, and a range of interpretations of what “autonomy” means. In developing this ICD, a balance has therefore been struck between the specific needs of the SAPIENT Middleware demonstration system and a desire to provide a flexible and extensible interface that could cope with unknown future sensor and processing modules in potential future iterations of the SAPIENT Middleware system beyond the current instantiation.

1.4 New Additions in Version 7

Following experimentation at the NATO Counter-Uncrewed Air Systems (C-UAS) Technical Interoperatbility Exercise (TIE) it was decided to move SAPIENT from using an XML (eXtensible Markup Language) message format to Google's Protobuf message format. Whilst XML was a human-readable format, Protobuf is a binary format, this will enable a reduction in the SAPIENT message size of approximately 60% which is a critical requirement in many defence Use Cases. Some structural changes to the ICD have been made to support the introduction of Protobuf, most notable use of enumerated fields.

Version 7 also introduces some changes in terminology. Version 6 of the ICD introduced effectors in addition to sensors (Automated Sensor Modules – ASMs), sensors and effectors are now both refered to as 'nodes' in the ICD. The fields 'sensorID' and 'sourceID' has been replaced with 'nodeID' and 'destinationID'. The data types of some fields have been changed (mostly from integers to strings), this mostly effects identifier (ID) fields; these fields are now Universally Unique Identifiers (UUID v4), or a Universally Unique Lexicographically Sortable Identifier (ULID). The ULID includes a date/time element which makes it easier to sort and is used where an ID may get updated regularly, for example, detections. The use of UUIDs and ULIDs in general will eliminate the need to pre-allocate the SAPIENT system with nodeIDs, and will prevent collision between IDs when SAPINET adopts a hierarchal architecture. The final significant change in this version is the introduction of velocity fields in the Detection messages.

There have also been some changes to fields in the ICD compared to earlier versions. Some fields that had been added as placeholders for future functions have generally been removed. Some fields have been renamed for clarity (e.g. status in both the AlertAck and TaskAck messages have been renamed to 'alert_status' and 'task_status'). Use of the term 'heartbeat' has been removed in favour of 'Status' to improve consistency in the document.

The 'destination_id' field has been moved to the top level of the message. This now implies the Registration Acknowledgment message in the .proto files is now empty. This is considered undesirable and thus an accepted/rejected flag has been added to the message.

There have also been some changes to the structure of the SAPIENT taxonomy with regards to electromagnetic (EM) emissions. EM emission is now a top-level class. The taxonomy is not considered a normative part of the ICD.

The SAPIENT Interface Management Panel (SIMP) would welcome feedback on how any of the changes introduced in this version have worked, and of any modifications provided to enable effective facilitation of these features.

1.5 Document Structure

This document is organised as follows:

Section 2 outlines the System Design.

Sections 3 and 4 give an overview of the interface messages and protocol.

Section 5 gives a detailed description of the messages during initialisation.

Section 6 gives a detailed description of the messages during normal operation.

Section 7 gives some example messages.

Appendix A contains the extendable Taxonomy of classes.

1.6 Intellectual Property Considerations

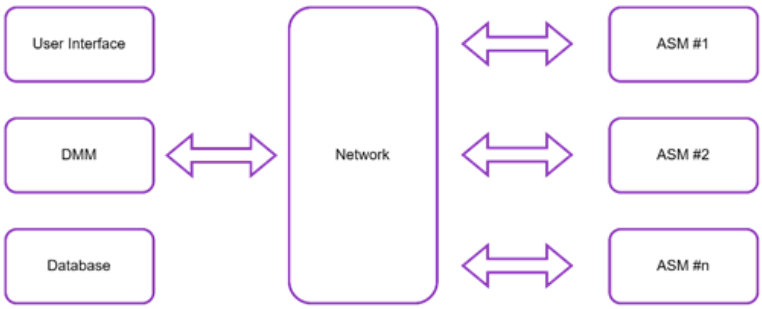
This Interface Control Document is Crown-owned Copyright. It has been generated through collaboration with QinetiQ as Lead Systems Integrator (LSI), various ASM and DMM suppliers from across the SAPIENT phases (2013-2019), and the members of the SAPIENT Interface Management Panel (SIMP). Each of them has agreed that they will not assert any essential patent rights (those which are in practice essential for the use of any part of the standard, as opposed to merely representing one of multiple practical implementations thereof), nor any rights in information they have contributed towards the standard.

This does not prevent any supplier from developing Intellectual Property in implementations of technology that connects to this interface.

It is the nature of an interface specification that it needs to describe fully all of the possible interactions across the interface: the DMM supplier in particular needs to understand the types of information it could receive from each of the candidate ASMs. Care has therefore been taken to generalise the message specifications wherever possible, and to anonymise the details of what the sensors can provide, so that there is no traceability from this document to any individual sensors' capabilities.

This Document is released and licensed under the terms of the Open Government Licence (see front page). Any contributions or requested amendments must be provided to Dstl under the Creative Commons Zero (CC-0) licence. Any information not provided to Dstl with the CC-0 licence will be protected appropriately and will not be used or reflected in this document.

2 System Overview



The SAPIENT Architecture

SAPIENT is intended to work at the “information level” rather than continually streaming raw data. The SAPIENT Middleware system architecture is based around a central database. A number of data agents manage the data-flow into the database. Each node communicates with a data agent to populate the database with declarations and detections, including as much information as the sensor modules are capable of providing. Although the SAPIENT philosophy is that C2 functionality is based on the ‘summary’ messages produced by the sensors, it is recognised that there are sometimes occasions where it is necessary to stream data from a sensor, and SAPIENT can provide the functionality to facilitate this. All ingest into the database is carried out by the data agents to ensure the data is valid. The data agent may monitor and if necessary limit data bandwidth. The nodes will communicate with the data agent using messages as described in sections 5 and 6.

The DMM performs higher-level fusion on the node messages and reasoning based on the data therein. The output of this reasoning is fused tracks and alerts for display on a GUI and tasking of nodes to provide further information such as imagery of objects of interest.

Similar to the nodes, the DMM communicates with an DMM data agent to pass its messages to the system. This populates the database with data describing fused tracks and alerts for passing to a GUI and tasks for nodes. The task messages are then forwarded to the relevant node via the data agent connected to the node. A separate set of database tables is used for DMM messages.

To provide scalability it is envisaged that each SAPIENT database/DMM combination will only work with a local set of sensors. Where multiple sets of sensors are deployed a hierarchy of SAPIENT systems could be deployed.

2.1 Network Interface

SAPIENT as a standard is intended to specify the content of the SAPIENT messages rather than constrain the means of communicating the messages between modules.

Different modules in the architecture communicate using TCP socket connections. As a result, any standard Internet technology such as Ethernet or Wifi that supports TCP may be used.

2.1.1 Protobuf Messages

This ICD defines a set of Protobuf messages for each component to populate the system, with its detections and declarations. Only valid Protobuf messages should be sent over the connections. It is intended that the autonomous behaviour of the system should be encapsulated within these Protobuf messages. Whilst both the ICD and .proto files describes the messages, the ICD will remain as the master definition of the messages.

The advantage of Protobuf is that it is readily extendable, and also is easy to adapt for the varying levels of information provided by each node. No individual node will be expected to populate all the fields; however, every node will be expected to populate the mandatory sub-set of these fields.

This ICD defines the following key messages:

- Registration
- Status Report
- Detection Report
- Alert
- Task

2.1.2 Message Framing / Structure

The protobuf binary format is not self delimiting: when two binary messages are concatenated, there is no way to tell where one message ends and the next message begins. To enable decoding of the protobuf messages when they are serialised over a continuous byte stream, such as a TCP connection, each message should have the length of the message in bytes added as a prefix to the message. This length should take the form of a 32-bit (4-byte) little endian number.

The protobuf messages are combined in a single “SAPIENTMessage” message. This contains information ((source) node ID, timestamp, and optionally, the destination node ID) that is common to all messages. This is followed by a single ‘One Of’ message content field; this contains the remaining fields for each message type.

2.1.3 Version of Standard String

The .proto file for the SapientMessage message contains a string representing the version of the standard that those .proto files suppose. This string is embedded as a protobuf file option. This is intended to be used for informational purposes (rather than as part of some formal version negotiation mechanism).

2.1.4 Additional Data

Additional data such as image snapshots may be needed to allow a system operator to respond appropriately to an alert. This should be referenced by URLs.

2.2 System Clock

All network devices shall synchronise to a central Network Time Protocol (NTP) Server. All modules will be expected to synchronise with this at regular intervals during system operation.

2.3 Behavioural Architecture

Under normal operation:

- Detection reports, Status Reports and Alerts are generated by the node and passed over the node interface to the Network.
- The DMM directly queries the database and performs processing to determine whether to raise an alert to an operator and/or task a node.
- When the DMM generates an alert that it wishes to raise up to the operator, it passes an Alert message to the Network, which populates the database, which in turn is queried by the UI.
- Most tasks and instructions will be generated autonomously by the DMM, and passed via the Network to the appropriate node.

Provision has been made for an operator to provide instructions from the UI, via the Network, to override the DMM in cases such as the control of an effector where a man-in-the-loop may be necessary.

The DMM is allowed direct access to the database for read but not for write. This ensures that the responsibility for the integrity of the database lies with the Middleware.

3 Interface Description Overview

3.1 Message Descriptions

All interaction between the nodes and the DMM will be via the Middleware hosted on the Network, and consists of these activities:

1. Initialisation;
2. Ongoing messages during normal operation;
3. (Optional) node Alerts during normal operation
4. Task messages from the DMM to individual nodes;

These will now be described in more detail.

3.2 High level process walkthrough

3.2.1 Initialisation

- Node connects as a client to the Middleware;
- Node sends Registration message to the DMM via the Middleware;
- The Middleware sends Registration Acknowledgement message to the node;
- Node sends initial Status message to the Middleware;
- The Middleware stores (all) messages in the database;
- The sensor nodes begin activity detection as per their default tasking (for those sensors which have defaults).

NB: If the DMM defines a task for an node, that task over-rides any node-generated default task previously defined.

3.2.2 Normal Operation (Sensor Nodes)

- ASM performs activity detection;
- ASM sends Detection messages as per current tasking to the Middleware;
- ASM sends regular Status messages to the Middleware;
- ASM optionally sends Alert messages to the Middleware (see below);
- Middleware stores messages in database;
- DMM queries the database, and reasons over/fuses the detections;
- DMM sends detection messages containing fused track points to the Middleware;
- DMM sends alert messages to the Middleware;
- DMM send task messages to ASMs via the Middleware;
- The Middleware stores (all) messages in database;
- UI queries the database to show the output from the DMM and/or ASMs.

3.2.3 Node Alerts during Normal Operations

- If a node wishes to inform the DMM that it is about to do something, the alert message can be used to inform or alert the DMM of this. An example of this might be a sensor deciding to do a detailed scan of an object or area.
- If the alert message merits a response from the DMM, the Alert Response message can used.
- If the DMM chooses to deny permission to carry out the action, the DMM will respond with an Alert Response message with 'status' field of value 'Reject'. The 'reason' field may be used to provide a strategy for retrying the action in future. (See section 3.8.1)

3.2.4 Control (Task) messages

- DMM sends SensorTask Message to node via the Middleware;
- Node sends acknowledgement of message to DMM via the Middleware – confirming acceptance or otherwise of tasking or command;
- Node changes behaviour as per command;
- Node returns to Normal Operation.

3.2.5 Node Shutdown

- Node sends Status message with "goodbye" tag to show impending loss of communication from this ASM;
- Node closes network connection to the Middleware.

3.2.6 Lost connection

It is the DMM and node module's responsibility to monitor connection the Middleware and attempt to reconnect if connection is lost. If a module is unable to connect to the Middleware, then it should attempt to retry every 10 seconds until it is successful.

If reconnection occurs over a short period of time (a few minutes), re-sending a Registration message may not be necessary. If a longer time has passed, the ASM should re-send a Registration message.

3.3 Initialisation

3.3.1 Registration Message Description

As part of the system initialisation, each node must produce a Registration message stating what its capabilities are. This will include:

- node type;
- supported detection modes;
- a list of what it is able to detect, track and/or classify;
- a declaration of error characteristics associated with the detections/tracks/classifications;
- a description of the contents of its Status message (see below);
- a list of how it is able to interact with the DMM – e.g. tasks it is able to accept.

The message will also include the performance implications of different detection modes. Performance can be specified in terms of geometric (location) error, detection performance and classification performance.

The Registration messages will contain declarations of node capability. The detail of the Registration message is given in section 5.1.

3.3.2 Registration Message Protocol

The Registration message shall be sent by the node to the Middleware before any other messages. No other messages shall be sent until a Registration Acknowledgement message or Error message is received in response. If successful, the Registration message only needs to be passed once, when the node connects or reconnects to the system.

The Middleware forwards the message to the DMM. The Middleware generates a Registration Acknowledgement message confirming the nodes existing node ID. The Middleware sends the acknowledgement to the ASM.

If the node does not receive a response within 30 seconds of sending the Registration message, the ASM should close the existing socket connection, open a new socket connection to the Middleware and send the message again.

The Registration message is stored in the database for use by the DMM and UI.

3.3.3 Location Information

The Registration message is used to declare the real-world coordinates system that the node will provide location information to the Middleware in all subsequent messages. Section 4.2 provides guidance on how location information should be provided.

3.3.4 Node ID

Each module instance must use a single, unique module identifier in all messages. This identifier should be a Universally Unique Identifier (UUID v4). The purpose of this identifier is to allow the middleware and DMM to identify the origin of messages. Registration messages, control messages and their acknowledgements are specific to an ASM and so ‘node_id’ is used in these messages. Status Report, Detection Report and Alert messages can be generated by ASMs or DMMs and so ‘node_id’ is used in these messages. The same identifier number shall be used across all messages.

The node_id is the ID of the source of the message. In all cases, the node shall use the node_id value sent in the initial registration message for all subsequent messages.

3.3.5 Initial Status message

As part of system initialisation, an initial Status Report message will be sent after the Registration message. This will indicate the initial state and optionally the physical location and field of view of the node. For ASMs providing detection messages in RangeBearing (Spherical) coordinates, a sensor location must be provided. For ASMs that provide detection messages in Cartesian coordinates, providing a sensor location is optional but can be useful for calibrating sensor location if the ASM does not have its own GPS or GPS accuracy is degraded.

3.4 Ongoing messages from ASMs during normal operation

During normal operation, ASMs provide Detection and Status messages. These follow the Protobuf schema in the .proto files which are available on request. The Protobuf fields are described further in section 6. The messages are stored in the database by the Middleware for the DMM and UI to read.

3.4.1 Status Messages

These are regular messages that must be produced by all nodes. This message will contain a time-stamp and the node's unique ID, and will provide the DMM with reassurance that the (sensor) node is still operating correctly, even in the absence of any valid detections. In addition, this message may provide additional information (as declared in advance in their initialisation script). This information could include for example, some of the following fields:

> Node location, field of view; coverage; node mode; obscuration polygon(s); rain detected; degraded sensor performance; remaining battery power; camera exposure time.

In addition to regular Status messages, one-off Status messages may be sent immediately on important changes. Examples of this might include the following:

> ASM tamper, ASM fatal error, ASM completely obscured

It is recommended that a regular Status message should be sent at least once every 60 seconds and not more than once a second. If no Status or Detection messages are received by the Middleware within a 60 second period, the Middleware will close the socket connection and wait for the ASM or DMM to re-establish communication.

At least the latest status information will be stored in the database. Storing some historical status information may be useful for diagnostic purposes, but to limit redundant data, this may be limited to where information has changed.

The Status message is described further in section 6.1

3.4.2 Detection Messages

These messages are output by each ASM when they wish to declare a detection of an object or other result (e.g. classification) to the wider system. The message declaration contains the sensor's unique ID, a timestamp, an object unique ID number, the real-world location of the detected object (at least 2D location, some sensors will be able to produce 3D locations and/or location error ellipses), and as much additional information about the detected object as the ASM can provide (in line with the declaration made in the Registration message). This could include for example the detection confidence level, track information, and/or classification information (with confidence levels). A URL to additional data such as an image snap-shot or point-cloud may also be provided.

The format of the messages from all of the ASMs shall be consistent with the given protobuf schema, however the actual content of the messages will potentially be different for each ASM, depending on its capabilities. No individual sensor module will be expected to fill in all the possible fields – each ASM will provide as much information as it can, as described in its registration message, and the DMM will be expected to reason across it and fuse it appropriately.

The Detection message is described further in section 6.2

3.4.2.1 Object ID

Object ID is an Unique Lexicographically-sortable Identifier (ULID) generated by an ASM that uniquely identifies an object. If an ASM is not capable of associating multiple detections of the same object then the object ID generated should be unique to each detection message.

When an object remains static in the field of view for a long time such as a vehicle parking up, it is assumed that it will merge into the background in time. If detected as such, it should be reported as “lostInView”. If supported, it may then be reported as an obscuration.

3.5 ASM Alert Messages during normal operation

Node generated Alert messsages provide a means for an node to report something that needs a response from the DMM. An example might be that a node wishes to autonomously change mode, but would like permission from the DMM. Modes requiring permission should include modeParameter type="PermissionToChange" value="Required" in the registration message. When requesting permission to change mode, the ‘description’ field in the Alert message should be used to to provide details of the request, e.g. “Requesting Scan3D”.

They can also be used to provide extra information directly from the node to the UI. In this case no response would be expected.

Finally, if the system includes sensors that can only provide alerts, such as door alarms, these can be reported using Alert Messages. Where a response is required an Alert Response message is used.

DMM generated Alert messages provide the means for the DMM to present alerts to the operator via the UI and may allow the operator to provide a response to the system.

The Alert message is described further in section 6.3

3.5.1 Message Protocol

As per other messages from nodes, Alert messages are stored in the database by the Middleware for the DMM to read.

Alert messages may also be additionally forwarded to the DMM via the Middleware to allow the DMM to provide an immediate response, such as when an node wishes to autonomously change mode, but would like permission from the DMM.

3.6 Task Messages

Task messages are sent by the DMM via the Middleware to task individual nodes. The overall philosophy used is that commands and filtering options are task-based, with an optional associated region defined in the task command. The particular message information will be very much dependent on the capabilities of the individual node, as described in the Registration message. However, the general form will be request ID, sensor ID, and a tasking request, for example

- In Region X look for threats of Type Y;

- Ignore all detections in Region Z;
- Look at Location (X,Y);
- Request a snapshot;
- Another sensor has detected something at location (X,Y), can you confirm the detection, track the object and classify the activity. This is effectively a cross-cue from one sensor to another;
- Your false alarm rate is unacceptably high (or your P(d) is unacceptably low) – adjust your thresholds accordingly.

SensorTask messages contain two sections:
1. A Region Definition related to the task;
2. A command or action related to the task.

Command based tasks tend to be short lived or instantaneous whereas Region ones tend to persist until another task is received.

Command based tasks such as 'Request Snapshot' do not require the region section to be populated. Similarly if a Region of Interest is defined for a node in the Region section of the message, the command section does not need to be populated.

The Task message is described further in 6.5, the region section in 6.5.3 and the command section in 6.5.4

3.6.1 Task Message Protocol

The DMM sends Task Messages to nodes via the Middleware.

The node responds to the Task message with one or more TaskACK acknowledgement messages during the lifetime of the task. The acknowledgement message states whether the node accepts or rejects the task and may include any additional data such as a URL to a snaphot. If a node cannot comply with a control message, it sends a TaskACK acknowledgement message with a description as to why in the 'reason' field of the message.

The Middleware forwards the Acknowledgement message back to the DMM and stores the response in the database.

When a task completes or is stopped by a control message, the node will revert to any previously defined task (such as the default one). If an ASM does not have a default task it will stop detecting until it receives a new task.

3.6.2 Sensor/Effector Modes

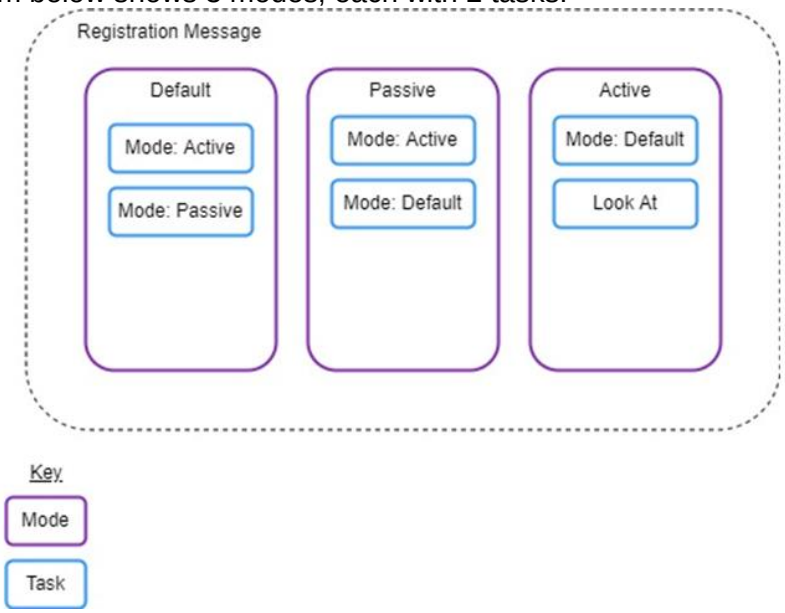
An ASM will operate in one or more sensor/effector modes. A sensor/effector mode is defined as a distinct way of operating. More than one mode may be requested from the DMM by providing a comma-separated list of modes.

A non-steerable sensor may operate in a single sensor mode. A steerable sensor is likely to have multiple modes that define how it will respond to tasking from the DMM. For example, a steerable camera could have one mode where it continually scans an area and another where it lingers at a number of fixed locations.

An effector will typically have a number of effecting modes that will be specific to its effector type. For example, a jamming effector may be capable of operating at a number of frequencies, with different waveforms and using antennas with different gains.

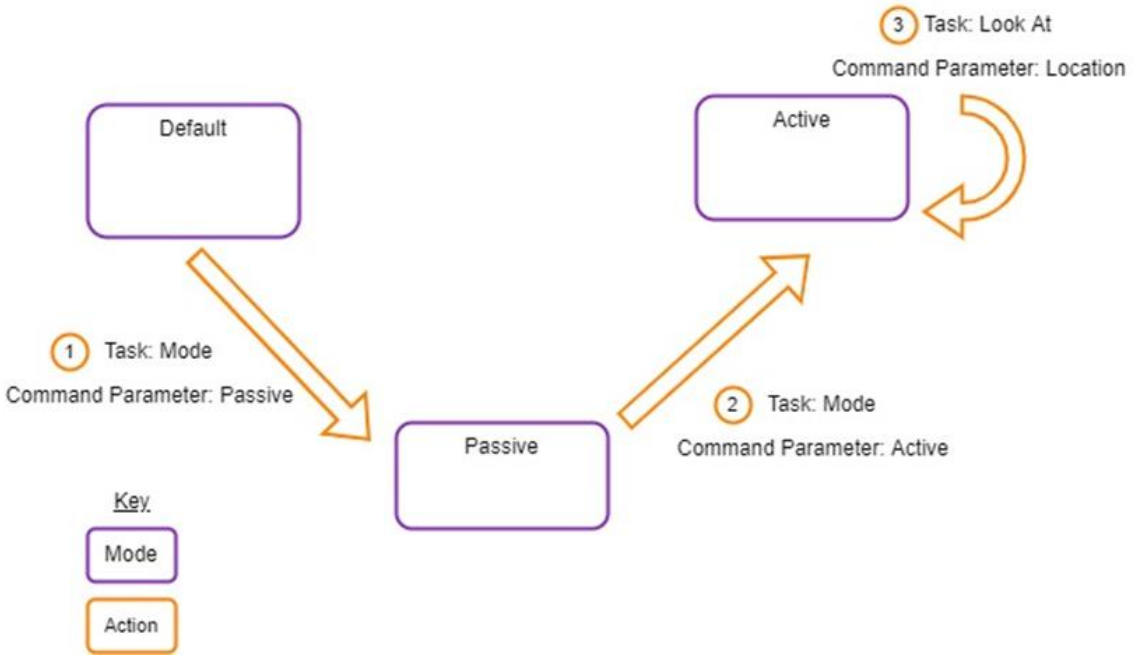
To achieve interoperability between an ASM and DMM, a set of pre-defined modes may be used to avoid each ASM defining modes independently. This approach will be application-specific, however, pre-defined mode sets may be added as an annex to this ICD to provide a preferred option for a specific project to consider. To be compliant with a pre-defined mode set, the ASM does not need to support all of the modes in the set.

The modes supported by an ASM shall be declared in the Registration message. The diagram below shows 3 modes, each with 2 tasks:



Example mode-task definition in a registration message

Modes can be thought of as a state machine, with changes occurring during tasking. The diagram below shows how an ASM's mode would change during tasking. The diagram assumes the modes/tasks shown in the previous diagram:



Example tasking as a state machine

3.6.3 Task Message 'Control' Field

The 'control' field in the Task message defines what action to perform on that task, (see section 6.5.2).

3.7 Acknowledgement Messages

Acknowledgement Messages are used to complete the handshake between the components of the system. These are only used in response to specific messages. There are three types of ACK message:

- Registration Acknowledgement from the DMM to the node that registration has occurred (see section 5.7);
- Task Acknowledgement from the node that it has received a sensor tasking message from the DMM (see section 6.6);
- Alert Acknowledgement from the DMM that is has received an alert message. (see 3.8 and 6.4)

3.8 Alert Acknowledgement Messages

Alert Response Messages are used where a response is required to an Alert Message. An example of this might be a sensor deciding to do a detailed scan of an object or area and it wants to ask permission from the DMM first. The DMM would send an Alert Response message with 'Acknowledge' or 'Reject' status field. Modes requiring permission should include modeParameter type="PermissionToChange" value="Required" in the registration message.

3.8.1 Reject Responses

Where an Alert Response Message has 'Reject' in the status field, the reason field can be used to allow the DMM to inform the ASM when it can try to send this request again. An empty reason field implies that this request should not be sent again. To allow the ASM to send this request again use 'Retry:' in the reason field, followed by the time interval and time units e.g. 'Retry:10,seconds'.

4 Guidance on using the Protobuf schema

4.1 General Guidance

A system potentially contains a wide range of sensor/effector types and capabilities. It is recognised that no one sensor will be expected to fill in all of the fields. Additional guidance is given in sections 5, 6 and Appendix A as to which fields are compulsory and which are optional.

4.2 Location Information

To support a variety of sensor types, the Middleware should accept location information in a number of different real-world coordinates systems. GPS and UTM coordinate systems should be supported for global locations (also referred to Cartesian locations). The Middleware should also support detections and status information in Spherical coordinates (range/bearing/elevation) relative to a known sensor location. Providing object locations in any of these formats is acceptable, depending on what the native format is of the individual ASM.

To keep this flexibility bounded, this will be limited to Cartesian (x,y,z coordinates, together with a corresponding reference frame, either UTM or GPS) or Spherical (Range-bearing-elevation). Within each of these, a number of coorindate systems and datums/units are provided as enumarations (e.g. LAT_LNG_DEG_M being latitude and longitude being provided in decimial degrees with the z coorindate being in meters).

There are a number of different types of locations provided in the Protobuf messages; (i.e. detections, sensor location, sensor field of view, areas of obscuration, regions). The following table provides guidance on which coordinates system to use for the different types of location. Where supported by an node, the coordinates system used shall be declared in the Registration message. This coordinates system shall be used consistently for all messages that contain location data of that type. Non-selected coordinates systems shall then be ignored entirely. For Cartesian coordinates, the node shall use the same reference frame (i.e. GPS or UTM) for all location types.

Location Type	**Coordinates System**
Detections	Any
Sensor Location	Cartesian
Sensor Field of View and Coverage	Any (Range-Bearing may be preferable for steerable sensors)
Areas of obscuration	Any
Regions	Any – Cartesian preferred because regions may be independent of sensors

Coordinate Systems for different types of location information

Any object sizes, speeds etc. reported shall also be given in real-world units. Internal frames of reference (e.g. camera-centric, measured in terms of pixels) should be calibrated and translated into real-world values (with associated errors if appropriate) before being reported.

The DMM shall task the individual nodes using the chosen coordinate system of the node as specified by the TaskDefinition section of the ASM Registration message.

All azimuth bearings shall be reported relative to north, Grid North should be used by default but magnetic or true north can be specified in the node's Registration message.

4.3 Location Protobuf fields

Location information is provided by one of the following Protobuf fields:

4.3.1 Location

The location message defines a single point in space in Cartesian form as follows:

Type	Field Name	Status	Description
double	x	M	Eastings in metres up to 2 decimal places
double	y	M	Northings in metres up to 2 decimal places.
double	z	O	Altitude of object in metres up to 2 decimal places - if omitted then object on ground-plane will be assumed. Altitude may be calculated by summing the relative value to the sensor and the known sensor altitude so that terrain data is not required.
double	x_error	O	Error of location in metres up to two decimal places
double	y_error	O	Error of location in metres up to two decimal places
double	z_error	O	Error of location in metres up to two decimal places
LocationCoordinateSystem enum	coordinate_system	M	Coordinate system used (see 4.3.1.1)
LocationDatum enum	datum	M	Datum used (see 4.3.1.2)

4.3.1.1 LocationCoordinate System Enumeration

The coordinate system and units used by the location. Each enumeration name shall be prefixed with LOCATION_COORDINATE_SYSTEM_. Such as:

0. UNSPECIFIED – Location coordinate system not defined
1. LAT_LNG_DEG_M – Latitude/Longitude in decimal-degrees with altitude in metres
2. LAT_LNG_RAD_M – Latitude/Longitude in radians with altitude in metres
3. LAT_LNG_DEG_F – Latitude/Longitude in degrees with altitude in feet
4. LAT_LNG_RAD_F – Latitude/Longitude in radians with altitude in feet
5. UTM_M – UTM coordinates with altitude in metres

4.3.1.2 Location Datum Enumeration

The datum used by the location. Each enumeration name shall be prefixed with LOCATION_DATUM.

0. UNSPECIFIED – Location datum not defined
1. WGS84_E – WGS84 Ellipsoid
2. WGS84_G – WGS84 Geoid

4.3.2 locationList

This is a list of location elements that define the boundary polygon of a region or sensor field of view

4.3.3 RangeBearing

The rangeBearing field defines a single point in space in Spherical form as follows:

Type	Field Name	Status	Description
double	elevation	O	Angle above/below horizontal in degrees up to 2 decimal places - if omitted then object will be assumed to be on the ground-plane
double	azimuth	O	Angle of detection relative to North - based on system wide concept of north
double	range	O	Distance from sensor to object in metres up to 2 decimal places
double	elevation_error	O	Error of location in metres up to two decimal places
double	azimuth_error	O	Error of location in metres up to two decimal places
double	range_error	O	Error of location in metres up to two decimal places
RangeBearingCoordinateSystem enum	coordinate_system	M	Coordinate system used (see 4.3.3.1)
RangeBearingDatum enum	datum	M	Datum used (see 4.3.3.2)

4.3.3.1 RangeBearing System Enumeration

The coordinate system and units used by the location. Each enumeration name shall be prefixed with RANGE_BEARING_COORDINATE_SYSTEM_. Such as:

0. UNSPECIFIED – Range bearing coordinate system not defined
1. DEGREES_M – Angles in decimal-degrees with range in metres
2. RADIANS_M – Angles in radians with range in metres

3. DEGREES_KM – Angle in degrees with range in kilometres
4. RADIANS_KM – Angle in radians with range in kilometres
5. DEGREES_F – Angle in degrees with range in feet
6. RADIANS_F – Angle in radians with range in feet

4.3.3.2 RangeBearingSystem Datum Enumeration

The datum used by the location. Each enumeration name shall be prefixed with RANGE_BEARING_DATUM_.

0. UNSPECIFIED – Range bearing datum not defined.
1. TRUE – True north
2. MAGNETIC – Magnetic north
3. GRID – Grid north
4. PLATFORM – North is defined as the heading of the platform carrying the node

4.3.4 RangeBearingCone

The rangeBearingCone field defines a Cone or field of view in Spherical form as follows:

Type	Field Name	Status	Description
double	elevation	O	Angle above/below horizontal in degrees up to 2 decimal places - if omitted then object will be assumed to be on the ground-plane
double	azimuth	O	Angle of detection relative to North - based on system wide concept of north
double	range	O	Distance from sensor to object in metres up to 2 decimal places
double	horizontal_extent	O	Horizontal extent angle of region in degrees up to 2 decimal places
double	vertical_extent	O	Vertical extent angle of region in degrees up to 2 decimal places
double	horizontal_extent_error	O	Error in horizontal extent angle

double	vertical_extent_error	O	Error in vertical extent angle
double	elevation_error	O	Error of location in metres up to two decimal places
double	azimuth_error	O	Error of location in metres up to two decimal places
double	range_error	O	Error of location in metres up to two decimal places
RangeBearingCoordinateSystem enum	coordinate_system	M	Coordinate system used (see 4.3.3.1)
RangeBearingDatum enum	datum	M	Datum used (see 4.3.3.2)

4.3.5 LocationOrRangeBearing

Some messages require that either a location or range bearing be provided. In these situations a protobuf OneOf field is used to provide the options. These are structured as follows:

- Location is the location being reported
- RangeBearing is the range bearing being reported

Some messages require location list or range bearing cones. In these situations a protobuf one of field is used. These fields are structured as follows:

- LocationList
- RangeBearingCone

4.4 Time Information

Note that all timestamps are transmitted in protobuf Timestamp (well-known type) format. It is recommended that if these need to be rendered as strings that they are rendered in UTC Zulu using ISO8601 format:

i.e. yyyy-MM-ddTHH:mm:ss.fffZ

Detection and Status messages should have sub-second accuracy, to ease the task of the DMM in cross-correlating them. Other messages only need specify to the nearest seconds i.e.: yyyy-MM-ddTHH:mm:ssZ

The Middleware will provide a network time reference, Time information shall be synchronised to this time reference.

4.5 Further Guidance on Protobuf fields

To keep the protobuf messages concise, it is valid to omit fields that are not mandated. This is equivalent to specifying 'not available' in the report lists in the initialisation message.

For some fields, such in as the Location or RangeBearing messages, a value of 0 is sometimes a real value. Because protobuf removes ‘0’ or null fields these will not get transmitted. Where a field is mandatory, and there is no value given, developers should considered the case of an actual ‘0’ value. To avoid confusion,

most enumerations in the protobuf files have their ‘0’ value set to ‘Not Specified’ or equivalent to ensure that a field is not missed accidentally.

4.5.1 Text Fields

Where Protobuf fields include text, it is assumed that the first letter will be capitalised.

4.5.2 Class and Behaviour Taxonomy

It is up to each ASM supplier to define the list of class types that they can classify. The Taxonomy that the Middleware should support will depend upon the system purpose. A basic taxonomy is defined in Appendix A to allow it to evolve without impacting the rest of the document.

4.6 Managing Real-Time System Performance

Whilst the Middleware has been designed to provide sufficient data throughput for the DMM and end-user real-time requirements, it is prudent to have the ability to manage data bandwidth if issues do arise.

If the DMM decides it is receiving too much data from one or more ASMs to process in real-time, then the DetectionThreshold and DetectionReportRate commands can be used to limit the data being sent across the system.

If the Middleware detects an issue that may not be apparent to the DMM, it can send an Alert Message to the DMM to appraise it of the issue so that it can then decide on the most appropriate course of action.

It is recommended that the TCP_NO_DELAY flag is set for all socket connections to minimise the latency through the system.

5 Message Detailed Description - Initialisation

This section describes the content of the messages used during initialisation.

The status column describes whether the field is Mandatory (M) or Optional (O) Optional. The type column indicates the present of a repeated (List) field through the use of square brackets ([]).

All SAPIENT messages have a ‘header’ of the time the message was sent and the node ID of the sender. The body of the message is stored in the ‘content’ field, which is one of the other SAPIENT messages defined in this document.

Type	Field Name	Status	Description
google.protobuf.Timestamp	timestamp	M	UTC time of message
string	node_id	M	The node ID of the sender of the message (UUID v4)
string	destination_id	O	The destination node for the message
oneof	content	M	The actual SAPIENT message

The OneOf in the content field contains the one of the following SAPIENT messages:

- Registration (see 5.1)
- RegistrationAck (see 5.7)
- StatusReport (see 6.1)
- DetectionReport (see 6.2)
- Alert (see 6.3)
- AlertAck (see 6.4)
- Task (see 6.5)
- TaskAck (see 6.6)
- Error (see 6.7)

5.1 Registration Message

On first connection to the Middleware, a node must send a Registration message. This will include a number of fields declaring the capabilities of the DMM. For any given field where the option Yes/No is provided in the message schema, a “No” response is equivalent to omitting the field altogether.

NB. Care should be taken to ensure that fields declared in the Registration message are consistent with those actually provided in the other message types.

Type	Field Name	Status	Description
NodeType enum[]	node_type	M	Node type defines the high-level category of node registering to allow the DMM to handle the ASM appropriately (see 5.1.1).
string	node_sub_type	O	This field provides a more detailed description of the node type (see 5.1.2).
string	icd_version	M	This is the ICD version the node/DMM implements. The value for this field should be taken from the 'Record of Changes' at the front of the ICD.
string	name	O	Optional name for individual instance of node e.g. Car Park Camera
string	short_name	O	2 digit abbreviation of name e.g. C1 for camera 1.
Capability[]	capabilities	O	List of node capabilities.See 5.2. These are used to provide guidance to DMMs on what modalities/functionality a node has
StatusDefinition	status_definition	M	See 5.3
ModeDefinition[]	mode_definition	M	See 0

5.1.1 NodeType Enumeration

The node type enumeration allows the modality or type of node that is registering to be specified. Node types should be specified if they are relevant to the node that is registering. Each enumeration type shall be prefixed with NODE_TYPE_.

0. UNSPECIFIED – no node type set
1. OTHER – used to represent a node type not included in this enumeration
2. RADAR – used to represent a node that uses/is a radar
3. LIDAR – used to represent a node that uses light to determine the range to a target
4. CAMERA – used to represent a node which contains/is a camera
5. SEISMIC – used to represent a node that uses ground vibrations
6. ACOUSTIC – a system based on acoustic signals, e.g. microphone
7. PROXIMITY_SENSOR – a system that can detect the local presence of objects, e.g. trip-wire
8. PASSIVE_RF – a system based on the passive receiption of electromagnetic radiation
9. HUMAN – a human acting as part of the SAPIENT system
10. CHEMICAL – a system that can detect chemical signatures
11. BIOLOGICAL – a system that can detect biological signature
12. RADIATION – a system that can detect radioactive signatures, e.g. Geiger counter

13. KINETIC – a system that has a physical effector, e.g. a missile
14. JAMMER – a radio frequency transmitter that disrupts other systems using power
15. CYBER – a system that uses network/communications protocols to detect or effect other systems

5.1.2 NodeSubType Enumeration

The node sub-type (optional) field allows additional, more detailed information about the node that is registering to be specified. This shall be a comma-separated list of detailed capabilitiues the node posses relevant for each 'Node Type' the node reports. For example, if a node reported node_type's of 'Camera' and 'Radar' it could report the following node_sub_type's; for Camera, it could report "Full Motion Video, Stereoscopic" if the node was capable of reporting distance to an object based on parallax between two apertures, and a continuious stream of images at a regular time interval. For the Radar, the node could report "Synthetic Aperture Radar, MTI" if the node is capable of forming an image via a synthetic aperture and also capable of reporting the motion of entities of interest.

5.2 Registration – capabilities

This is a list of capabilities that an node can provide such as type of Electronic Signal Detection. It has the following fields:

Type	Field Name	Status	Description
string	category	M	The category of field to report. Eg. ESM, Jammer
string	type	M	Description of the capability e.g. Maximum Transmit Power
string	value	O	Value of this parameter e.g. 50.
string	units	O	A description of the units or valid values that will be reported e.g.dB

If these fields can be reported on an ongoing basis, they can be reported as part of the status list in the Status Report. See 6.1.9.

5.3 Registration – StatusDefinition

The heartbeat definition section of the Registration message defines which fields will be present in Status Report messages generated by the node and what format and units the fields will use.

Type	Field Name	Status	Description
Duration	status_interval	M	Time interval between regular 'heartbeat' Status Report messages being issued by the node. See 5.3.1
LocationType	location_definition	O	Define the Coordinates system to use for sensor location. E.g. UTM or GPS. NB. RangeBearing is not a valid value for sensorlocation
LocationType	field_of_view_definition	O	If providing sensor field of view, define the Coordinates system and units to use.
LocationType	coverage_definition	O	If providing sensor coverage, define the Coordinates system and units to use.
LocationType	obscuration_definition	O	If providing sensor obscuration regions, define the Coordinates system and units to use.
StatusReport[]	status_report	O List	Definition of additional fields reported by the ASM in its Status Report messages

5.3.1 Registration - Duration

This states the Time interval between regular 'heartbeat' Status Report messages being issued by the node. If a Status Report is not received within this interval, the DMM can assume this node has been disabled, turned off or lost communication.

Type	Field Name	Status	Description
TimeUnits enum	units	M	Typically 'SECONDS'. See 5.3.2
int32	value	M	Typically between 1 and 60

5.3.2 TimeUnits Enumeration

The time units enumeration allows units of time being used in a duration to be specified. Each enumeration shall be prefixed with TIME_UNITS_.

0. UNSPECIFIED – time units not specified
1. NANOSECONDS – used to express a duration in nano-seconds
2. MICROSECONDS – used to express a duration in micro-seconds
3. MILLISECONDS – used to express a duration in milli-seconds
4. SECONDS – used to express a duration in seconds
5. MINUTES – used to express a duration in minutes
6. HOURS – used to express a duration in hours
7. DAYS – used to express a duration in days

5.3.3 Registration - locationType

This defines the format and units for each field that can provide location information. Whilst it is valid for an node to provide some fields in Spherical (RangeBearing) coordinates and others in Cartesian (UTM, GPS) coordinates, it is recommended that UTM and GPS coordinates systems are not both used within the same node.

LocationDatum is used specify coordinate system datum in use, typically WGS84.

RangeBearingDatum is used to specify North datum in use, typically grid.

Type	Field Name	Status	Description
oneof	coordinates_oneof	M	Either Location or RangeBearing Coordinate system and units being reported
oneof	datum_oneof	M	Either the Location or RangeBearing Datum being reported
string	zone	O	If using UTM this is the, UTM Zone e.g. 30U, otherwise omit.

5.3.4 Registration - VelocityType

This defines the format and units for each field that can provide velocity information. Whilst it is valid for an ASM to provide some fields in Spherical (RangeBearing) coordinates and others in Cartesian (UTM, GPS) coordinates, it is recommended that UTM and GPS coordinates systems are not both used within the same ASM.

Type	Field Name	Status	Description
one of	velocity_units_oneof	M	Specify how velocity is provided by specifying the coordinates system type and units in use. Choose one of: GHCVelocityUnits RYPVelocityUnits RAEVelocityUnits ENUVelocityUnits SHPVelocityUnits
oneof	datum_oneof	M	Either the Location or RangeBearing Datum of the velocity being reported
string	zone	O	If using UTM this is the, UTM Zone e.g. 30U, otherwise omit.

5.3.5 Registration – GHCVelocityUnits

Specify units when GHCVelocity is in use

Type	Field Name	Status	Description
SpeedUnits enum	ground_speed_units	O	Units that ground speed is reported in
AngularUnits enum	angular_units	O	Units that heading are reported in
SpeedUnits enum	climb_rate_units	O	Units that climb rate is reported in

5.3.6 Registration – RYPVelocityUnits

Specify units when RYPVelocity is in use

Type	Field Name	Status	Description
AngularVelocityUnits enum	angular_units	O	Units that yaw and pitch rates are reported in
SpeedUnits enum	range_rate_units	O	Units that range rate is reported in

5.3.7 Registration – RAEVelocityUnits

Specify units when RAEVelocity is in use

Type	Field Name	Status	Description
AngularVelocityUnits enum	angular_units	O	Units that azimuth and elevation rates are reported in
SpeedUnits enum	range_rate_units	O	Units that range rate is reported in

5.3.8 Registration – ENUVelocityUnits

Specify units when ENUVelocity is in use

Type	Field Name	Status	Description
SpeedUnits enum	east_north_rate_units	M	Units that velocity is reported in for east and north axis
SpeedUnits enum	up_rate_units	O	Units that velocity is reported in for up axis

5.3.9 Registration – SHPVelocityUnits

Specify units when SHPVelocity is in use

Type	Field Name	Status	Description
AngularUnits enum	angular_units	O	Units that heading and pitch angles are reported in
SpeedUnits enum	speed_units	O	Units that speed is reported in

5.3.10 Registration – SpeedUnits Enumeration

This specifies the units of the scalar speed component of velocity being reported. Each enumeration shall be prefixed with SPEED_UNITS_.

0. UNSPECIFIED – speed units not defined
1. MS – meters per second
2. KPH – kilometres per hour
3. MPH – miles per hour
4. FS – feet per second

5.3.11 Registration – AngularUnits Enumeration

This specifies the units for angles. Each enumeration name shall be prefixed with ANGULAR_UNITS_.

0. UNSPECIFIED – angular units not defined
1. RADIANS – angular units in radians
2. DEGREES – angular units in degrees

5.3.12 Registration – AngularVelocityUnits Enumeration

This specifies the units for rates of changes of angles. Each enumeration shall be prefixed with ANGULAR_VELOCITY_UNITS_.

0. UNSPECIFIED – angular velocity units not defined
1. ANGULAR_VELOCITY_UNITS_RS – angular velocity units in radians per second
2. ANGULAR_VELOCITY_UNITS_DS - angular velocity units in degrees per second

5.3.13 Registration - StatusReport

This lists the fields that will be included in the Status Report message. Only fields relevant to the particular node should be included; all others should be omitted. So, for example, a radar sensor would not report on its exposure or white balance, and a mains-powered sensor would not report battery level. A list of typical values is provided in 5.3.15.

Type	Field Name	Status	Description
StatusReportCategory enum	category	M	The category of field to report. This maps to the other fields of the Status Report message i.e. sensor, power, mode, status (see 5.3.14)
string	type	M	The type or name of the information being provided
string	units	O	A description of the units or valid values that will be reported
bool	on_change	O	If True, only report when the value changes. If False, always report it

5.3.14 StatusReportCategory Enumeration

The status report category enumeration allows category of a heartbeat report to be specified. Each enumeration shall be prefixed with STATUS_REPORT_CATEGORY_.

0. UNSPECIFIED – status report category not defined
1. SENSOR – means a report about the sensor's performance
2. POWER – means a report about the node's current power situation
3. MODE – means a report about the node's current mode
4. STATUS – means status fields can be reported

5.3.15 Registration – StatusReport – Typical Values

Below is a list of typical entries in the heartbeat report list section of the Registration message. The type and units fields should be consistent with those provided in the Status Report. (See 6.1.9)

Category	Type	Units
sensor	sensorLocation	(BLANK)
sensor	fieldOfView	locationList or rangeBearingCone
sensor	coverage	locationList or rangeBearingCone
sensor	obscurations	locationList or rangeBearingCone
power	status	OK, Fault
power	level	percentage
mode	mode	Default, Others as defined in registration message
status	InternalFault	(BLANK)
status	ExternalFault	(BLANK)
status	Illumination	Bright, Dark, Normal
status	Weather	text
status	Clutter	Low, Medium, High
status	Exposure	F Stop
status	MotionSensitivity	probability
status	PTZStatus	Moving, Stopped
status	PD	probability
status	FAR	probability

5.4 Registration - modeDefinition

Type	Field Name	Status	Description
string	mode_name	M	Name of mode e.g. Default, Follow
string	mode_description	O	Free text description
Duration	settle_time	M	See 5.3.1
Duration	maximum_latency	O	See 5.3.1
ScanType enum	scan_type	O	Fixed, Scanning, Steerable, See 5.4.2 If omitted will assume Fixed.
TrackingType enum	tracking_type	O	None, Tracklet, Track, TrackWithReID. See5.4.3. If omitted will assume none
Duration	duration	O	See 5.3.1
ModeParameter[]	mode_parameter	O List	See 5.4.4
DetectionDefinition	detection_definition	M	See 5.5
TaskDefinition[]	task	M	See 5.6
ModeType enum	mode_type	M	Temporary, Permanent

5.4.1 ModeType Enumeration

This defines the mode type. If omitted will assume permanent. Each enumeration type shall be prefixed with MODE_TYPE_.

0. UNSPECIFIED – means mode type is not defined.
1. PERMANENT - means that the noide will stay in this mode until tasked otherwise.
2. TEMPORARY - means that the node will revert to it's previous mode after this mode is finished.

5.4.2 ScanType Enumeration

This defines the sensor field of view behaviour. If omitted will assume fixed. Each enumeration type shall be prefixed with SCAN_TYPE_.

0. UNSPECIFIED – means scan type not specified
1. FIXED - means a field of view that does not change and the all areas are detected at all times e.g. a fixed video sensor.
2. SCANNING - means that the sensor will only be detecting over part of its field of view at any time e.g. a rotating radar.
3. STEERABLE - means that the sensor can be steered to point at a location e.g. a Pan-Tilt-Zoom Camera.

5.4.3 TrackingType Enumeration

This defines the ASM tracking capabilities. If omitted will assume none. Each enumeration type shall be prefixed with TRACKING_TYPE_.

0. UNSPECIFIED – means the tracking type has not been specified
1. NONE - means there is no data association between detections of the same object. A unique objectID is generated by the ASM for each detection.
2. TRACKLET - means the ASM attempts to maintain the objectID between detections of the same object but no attempt is made to join broken tracks.
3. TRACK - means that the ASM applies a tracking algorithm to allow consistent objectID even behind occlusions.
4. TRACK_WITH_RE_ID - means that the ASM can provide tracks that can be re-associated based on features somehow.

5.4.4 Registration - modeDefinition – modeParameter

ModeParameter is a list of descriptive parameters for the mode such as dynamic behaviour that is not otherwise reported. These can be useful for the DMM to determine the level of Autonomy within the node. This is particularly relevant to scanning or steerable sensors that can alter their behaviour such as following a target or viewing multiple zones

Type	Field Name	Status	Description
string	type	M	e.g. SelfAdaptation
string	value	M	e.g. ROI, Range

5.5 Registration – detectionDefinition

This lists the fields that will be included in the Detection message whilst in this mode. Only fields relevant to the particular ASM should be included; all others should be omitted.

Type	Field Name	Status	Description
LocationType	location_type	M	Location units to be used by detection messages as per 5.3.3.
GeometricError	geometric_error	O	See 0
DetectionReport[]	detection_report	O List	See 5.5.2
DetectionPerformance[]	detection_performance	O List	See 0
DetectionClassDefinition[]	detection_class_definition	O	See 0
BehaviourDefinition[]	behaviour_definition	O List	See 5.5.10
VelocityType	velocity_type	O	See 5.3.4

5.5.1 Registration – detectionDefinition – GeometricError

This is a list of location error characterisations to allow the DMM to understand the detection performance of the ASM.

Type	Field Name	Status	Description
string	type	M	e.g. Standard Deviation
string	units	M	e.g. metres
string	variation_type	M	e.g. Linear with Range, Squared with Range
PerformanceValue[]	performance_value	O List	A list of types and values

5.5.2 Registration – detectionDefinition – detectionReport

This lists the fields that will be included in the Detection message. Only fields relevant to the particular ASM should be included; all others should be omitted. A list of typical values is provided in 5.5.4

Type	Field Name	Status	Description
Detection ReportCategory enum	category	M	The category of field to report. This maps to the fields of the Detection message i.e. detection, track, object
string	type	M	The type or name of the information being provided
string	units	M	A description of the units or valid values that will be reported
bool	on_change	O	If True, only report when the value changes. If omitted or False, always report it

5.5.3 DetectionReportCategory Enumeration

This defines the type of detection report category being reported. Each enumeration type shall be prefixed with DETECTION_REPORT_CATEGORY_.

0. UNSPECIFIED – detection report category not specified
1. DETECTION – used to supply detection fields
2. TRACK – used to supply track fields
3. OBJECT – used to supply object fields

5.5.4 Registration – detectionDefinition – detectionReport Typical Values

Below is a list of typical entries in the detection report list section of the Registration message and the associated field in the Detection message. The type and units fields should be consistent with those provided in the Detection Report. (See 6.1.10)

Category	Type	Units	DetectionReport field
detection	confidence	probability	detectionConfidence
track	confidence	probability	trackInfo, ‘confidence’
track	speed	m-s	trackInfo, ‘speed’
track	az	degrees	trackInfo, ‘az’ (Heading)
track	dR	metres	trackInfo, ‘dR (Object Change in Range in metres’)
track	dAz	degrees	trackInfo, ‘dAz’ (Object Change in Heading)
track	predictedLocation	geo	predictedLocation
track	predictionTimestamp	UTC	predictionTimestamp
object	dopplerSpeed	m-s	objectInfo, ‘dopplerSpeed’
object	dopplerAz	degrees	objectInfo, ‘dopplerAz’
object	majorLength	metres	objectInfo, ‘majorLength’
object	majorAxisAz	degrees	objectInfo, ‘majorAxisAz’
object	minorLength	metres	objectInfo, ‘minorLength’
object	height	metres	objectInfo, ‘height’
object	colour	none	colour
object	state	none	state

5.5.5 Registration – detectionDefinition – PerformanceValue

A list of performance measures useful to the DMM - only provide if available.

Type	Field Name	Status	Description
string	type	M	Type of performance measure being defined
string	units	M	Units of performance measure being defined
string	unit_value	M	Typical value of the units of performance measure being defined
string	variation_type	O	Variation description of performance measure being defined

5.5.6 Registration – detectionDefinition – detectionClassDefinition

Type	Field Name	Status	Description
ConfidenceDefintion enum	confidence_definition	O	Single Class, Multiple Class. Omit if not providing confidence Single Class - only provide confidence for the most likely class. Multiple Class - provide confidence for all classes
PerformanceValue[]	class_performance	O List	See 5.5.6
ClassDefinition[]	class_definition	O List	See 5.5.8

5.5.7 ConfidenceDefinition Enumeration

This defines the type of confidence being reported. Each enumeration type shall be prefixed wth CONFIDENCE_DEFINITION_.

0. UNSPECIFIED – used to support default values in protobuf
1. SINGLE_CLASS – used to denote confidence supplied against the most likely classification only
2. MULTI_CLASS – used to denote confidence supplied against all possible classifications

5.5.8 Registration – DetectionDefinition – ClassDefinition

This defines the structure as to how object classes and sub-classes are provided for this ASM. Sub-classes allow a hierarchy to more narrowly define an object. All classes must be mutually exclusive and conform to the Taxonomy in A.1.

Type	Field Name	Status	Description
string	type	M	Type of class being defined
string	units	O	Units of the confidence of the class being reported
SubClass[]	sub_class	O	Definitions of any subclasses

Definitions of any child sub-classes make use of the SubClass message shown below:

5.5.9 Registration – DetectionDefinition – SubClassDefinition

Type	Field Name	Status	Description
string	type	M	Type of class being defined
string	units	O	Units of the confidence of the class being reported
int32	level	M	The level of the subclass (top level class would be 0, the first sub-class would be 1, etc.)
SubClass[]	sub_class	O	Definitions of any subclasses

5.5.10 Registration – DetectionDefinition - BehaviourDefinition

This defines the structure as to how object behaviour or activity is provided for this ASM. All behaviours must conform to the Taxonomy in A.2.

Type	Field Name	Status	Description
string	type	M	Type of behaviour being defined
string	units	O	Units of the confidence of the behaviour being reported

5.6 Registration – taskDefinition

Type	Field Name	Status	Description
int32	concurrent_tasks	O	The number of different tasks that can run at a time in this mode
RegionDefinition	region_definition	M	See 5.6.1
Command[]	command	O List	See 5.6.8

5.6.1 Registration – taskDefinition - regionDefinition

Type	Field Name	Status	Description
RegionType enum[]	region_type	M List	See 5.6.2
Duration	settle_time	O	Time to settle to normal performance in this mode
LocationType[]	region_area	M List	Location type to use for region definition
ClassFilterDefinition[]	class_filter_definition	O List	See 5.6.3
BehaviourFilterDefinition[]	behaviour_filter_definition	O List	See 5.6.7

5.6.2 RegionType Enumeration

This specifies the type of confidence being defined. Each enumeration type shall be prefixed with REGION_TYPE_.

0. UNSPECIFIED – used to support default values in protobuf
1. AREA_OF_INTEREST – used to show a region which is focused on by the ASM
2. IGNORE – used to show a region which is ignored by the ASM
3. BOUNDARY – used to show a region which defines the boundary of the area of operations

5.6.3 Registration – regionDefinition - classFilterDefinition

Type	Field Name	Status	Description
FilterParameter[]	filter_parameter	O List	See 5.6.4
SubClassFilterDefinition[]	sub_class_definition	O List	List of sub-class filters See 0
string	type	M	Class type to filter on e.g. “Human”, “Vehicle”

5.6.4 Registration – RegionDefinition - FilterParameter

This is a list of class parameters the ASM can filter on. Typically confidence is the main characteristic to filter on and the operator is typically ‘Greater Than’.

Type	Field Name	Status	Description
string	parameter	M	Name of the parameter that can be filtered
Operator enum []	operators	M	Operators that can be used during filtering (see 5.6.5)

5.6.5 Operator Enumeration

This specifies the type of operator that can used by the filter. Each enumeration type shall be prefixed with OPERATOR_.

0. UNSPECIFIED – operator not specified
1. ALL – filter on all of this parameter
2. GREATER_THAN – filter on values of the parameter greater than a given value
3. LESS_THAN – filter on values of the parameter less than a given value
4. EQUALS – filter on values of the parameter equal to the specified value

5.6.6 Registration – regionDefinition - subClassFilterDefinition

Type	Field Name	Status	Description
FilterParameter[]	filter_parameter	O List	See 5.6.4
string	type	M	Sub-Class type to filter
int32	level	M	Sub class hierarchy level that this type is at. Typically 1-3.
SubClassFilterDefinition[]	sub_class_definition	O	List of sub-class filters

5.6.7 Registration – regionDefinition - behaviourFilterDefinition

Type	Field Name	Status	Description
FilterParameter[]	filter_parameter	O List	See 5.6.4
string	type	O List	Behaviour type to filter on e.g. “Running”

5.6.8 Registration – taskDefinition – command

The command list in the Registration message gives the list of commands that an node supports, including the default commands that all nodes should provide. Some examples are provided below:

Type	Field Name	Status	Description
string	name	M	e.g. Request, DetectionThreshold, Mode. See list at 6.5.4
string	units	M	Valid Values for this command e.g. Heartbeat
Duration	completion_time	M	e.g. 1
CommandType enum	type	M	Type of task. Should match the type field in 5.6.9

5.6.9 Task Commands Enumeration

Below is a list of commands that an node may typically support. Further commands may be defined in the Registration message. The commands shall be interpreted differently depending on whether the node is a sensor or an effector. Each enumeration type shall be prefixed with COMMAND_TYPE_.

Enum	Type	Units	Description for Sensor Node	Description for Effector Node
0	UNSPECIFIED	n/a	Command type not defined	
1	REQUEST	Registration	Request the node sends a Registration message.	
1	REQUEST	Reset	Request the node ‘reboots’ itself.	
1	REQUEST	Status	Request the node sends a Status Report message.	
1	REQUEST	Arm	Request the ASM sets the	Request the node arms

			sensor into the currently-selected mode(s)	the effector with the currently-selected mode(s), ready for action
1	REQUEST	Start	Request the ASM starts sending Detection and Alert messages after a previous ‘Stop’ request.	Request the node starts the effector
1	REQUEST	Stop	Request the ASM stops sending Detection and Alert messages until told to start again.	Requests the node stops the effector
1	REQUEST	Take Snapshot	Task a sensor to take a snapshot. The URL to the snapshot should be provided in the SensorTaskACK message	Not used
2	DETECTION_THRESHOLD	Auto, Low, Medium, High	Allows the DMM to adjust the ASMs detection threshold or return it to its own ‘Auto’ threshold.	Not used
3	DETECTION_REPORT_RATE	Auto, Low, Medium, High	Allows the DMM to tell the ASM to report more or less detections or return it to its own ‘Auto’ rate.	Not used
4	CLASSIFICATION_THRESHOLD	Auto, Low, Medium, High	Allows the DMM to tell the ASM to be sensitive when classifying detections or return it to its own ‘Auto’ rate.	Not used

5	MODE_CHANGE	Default, Others as defined in registration message e.g. ‘Follow’, comma-separated if multiple simultaneous modes required	Allows the DMM to tell the node to change to a different operating mode(s)
6	LOOK_AT	locationList, rangeBearingCone	Allows the DMM to task a steerable node to look at a particular location.

5.7 Registration Acknowledgement Message

Upon receipt of a Registration message, the Middleware generates a Registration Acknowledgement message confirming the node’s existing node ID.

If the node does not receive a response within 30 seconds of sending the Registration message, the node should close the existing socket connection, open a new socket connection to the Middleware and send the message again. The ‘destination_id’ field at the start of section 5 shall be populated for the Registration Acknowledgement message. The Registration Acknowledgement message has the following format:

Type	Field Name	Status	Description
boolean	acceptance	M	This field shall take a value of 0 if the registration is rejected or a value of 1 if the registration is accepted.
string	ack_response_reason	O	This field can describe the reason for rejecting the Registration.

6 Message Detailed Description – Normal Operation

This section describes the content of the messages used during normal operation. The status column describes whether the field is Mandatory (M), or Optional (O); repeated (List) values are indicated with [] in the Type column.

All SAPIENT messages have a 'header' of the time the message was sent and the node ID of the sender. The body of the message is stored in the 'content' field, which is one of the other SAPIENT messages defined in this document.

Type	Field Name	Status	Description
google.protobuf.Timestamp	timestamp	M	UTC time of message
string	node_id	M	The node ID (UUID v4) of the sender of the message
string	destination_id	O	The node ID (UUID v4) of the destination of the message
oneof	content	M	The actual SAPIENT message

6.1 Status Report Message

This message is sent regularly to give the DMM information about the node operating mode, and to provide confidence that it is still working correctly (or conversely that performance can be expected to degrade). Only fields relevant to the particular node should be included; all others should be omitted. So, for example, a radar sensor would not report on its exposure or white balance, and a mains-powered sensor would not report battery level.

Type	Field Name	Status	Description
string	report_id	M	ULID of this report
System enum	system	M	Overall status of sender: OK, Warning, Error, Tamper, GoodBye. Node sends good bye to close the communication cleanly in case it needs to shut down and reboot. See 6.1.1
Info enum	info	M	Unchanged, New. A flag to tell the DMM if the information in this message is new or unchanged. See 6.1.2
string	active_task_id	O	ID of the task being performed by the ASM
string	mode	O	Name of the mode the ASM is currently in, as defined in Registration message. If a fatal error has occurred then report "Failed" here.
Power	power	O	See 6.1.4

Location	sensor_location	O	See 6.1.5
LocationOrRangeBearing	field_of_view	O	See 6.1.6
LocationOrRangeBearing	coverage	O	See 6.1.7
LocationOrRangeBearing[]	obscuration	O	See 6.1.8
Status[]	status_value	O	See 6.1.9

6.1.1 System Enumeration

This specifies the status of the node. Each enumeration name shall be prefixed with SYSTEM_.

0. UNSPECIFIED – the status of the node is unknown
1. OK – the node is working normally
2. WARNING – something has gone wrong with the node, but it can continue to operate
3. ERROR – there is an issue with the system
4. TAMPER – the system has been tampered with
5. GOODBYE – the system is about to disconnect

6.1.2 Info Enumeration

This specifies the type information contained in the Status Report. Each enumeration name should be prefixed with INFO_.

0. UNSPECIFIED – info not defined
1. NEW – new information is available in this Status Report
2. UNCHANGED – no new information is contained in this Status Report

6.1.3 Status Report – Mode

Mode is an optional field reported where a sensor provides multiple modes or states. E.g. Transition, Stopped, Failed, Default, 3D Scanning, Scan and Search.

If omitted, then ‘Default’ mode will be assumed.

If a fatal error has occurred, then ‘Failed’ should be reported.

Normally, the node will work in 'Default' mode but the DMM can request it changes into any other mode specified in the node Registration message. When a sensor is "on the move" and not detecting it should report as ‘Transition’. If an ASM has received a ‘Stop’ Request command, it should report ‘Stopped’.

6.1.4 Status Report - Power

This is used to report the status of the node power supply if available.

Type	Field Name	Status	Description
string	source	O	Mains or Battery
string	status	O	OK, Fault
int	level	O	Battery level, typically as a percentage e.g. 50

6.1.5 Status Report – Sensor Location

Sensor location is an optional field that reports the current geographical location of the sensor. With a fixed sensor, this may be reported once during initialisation although typically it is reported in every status report to aid debugging. Location is provided as a <location> element with optional fields for altitude and error (eX, eY, eZ)

6.1.6 Status Report – Field Of View

Sensor Field of View is an optional field that reports the current field of view of the sensor. With a fixed sensor, this may be reported once during initialisation. For steerable sensors, this will be reported regularly.

This should either be reported as a 'RangeBearingCone' proto message (see 4.3.3) or a 'LocationList' proto message (see 4.3.2). For a location list, points should be in order around the boundary of the field of view polygon.

6.1.7 Status Report – Coverage

Sensor Coverage is an optional field that reports the extent of the area a sensor can cover but not necessarily all at once. Typically, this will only be reported for scanning or steerable sensors. For example, a Pan Tilt Zoom camera may have 360degree coverage but only a field of view of 45degrees.

This should either be reported as a 'RangeBearingCone' proto message (see 4.3.3) or a 'LocationList' proto message (see 4.3.2). For a location list, points should be in order around the boundary of the coverage polygon.

6.1.8 Status Report – Obscuration

Obscuration is an optional field that can define one or more regions that the sensor can report as being unable to detect within. This refers to the region beyond a static obstacle, rather than the region in the field of view beyond a moving target. (If the DMM wants to know what region is obscured behind a moving target, it will be able to calculate the region using the reported object size and location in the detection report).

This should either be reported as a 'RangeBearingCone' proto message (see 4.3.3) or a 'LocationList' proto message (see 4.3.2). For a location list, points should be in order around the boundary of the obscuration polygon.

6.1.9 Status Report – Status

The 'Status' element of the status message provides a way for nodes to report conditions that may affect the performance of this or other nodes. The 'Level' field indicates the severity of the condition.

NB If a fatal error has occurred then report "Failed" under sensor mode as per 6.1.10.

The ‘status_type’ and ‘status_value’ fields should be consistent with the ‘status’ ‘type’ and ‘unit’ fields defined in the Registration message. (See 5.3.14).

For example, the following table shows the values in a Status field to indicate that rain has been detected that does not affect this sensor but may affect other sensors.

status_level	INFORMATION_STATUS
status_type	Weather
status_value	Rain

The following table shows the values in a Status field to indicate that the performance of the node has been degraded by external conditions e.g. rain snow, but it is still able to perform its current tasking

status_level	WARNING_STATUS
status_type	ExternalFault
status_value	n/a

The following table shows the values in a Status field to indicate that the performance of the node has been degraded by external conditions e.g. illumination, and so it is unable to perform its current tasking:

status_level	ERROR_STATUS
status_type	ExternalFault
status_value	Illumination

The ‘type’ field may be one of the following but is not limited to this list:

InternalFault, ExternalFault, Illumination, Weather, Clutter, Exposure, MotionSensitivity, PTZStatus, PD, FAR. (See also 5.3.14)

Type	Field Name	Status	Description
StatusLevel enum	status_level	O	Error, Warning, Information, Sensor (see 6.1.10)

string	type	M	See examples above
string	value	O	See examples above and in 5.3.14

6.1.10 StatusLevel Enumeration

This specifies the severity of the status condition being reported. Each enumeration name shall be prefixed with STATUS_LEVEL_.

0. UNSPECIFIED – status level not defined
1. SENSOR_STATUS – information about the current sensor state.
2. INFORMATION_STATUS – environmental condition have been detected that might affect other sensors.
3. WARNING_STATUS – performance is degraded.
4. ERROR_STATUS – something is severely affecting capability.

6.2 Detection Report Message

This message is sent by the ASM to report each detection (or, optionally, group of detections or track). Track data is an optional element used to provide the DMM with more information about the movement of an object, where available.

Where sensor performance allows, classification information should also be included for each detected object.

Type	Field Name	Status	Description
string	report_id	M	Unique identifier (ULID) of this message. The date/time component of the ID is updated with each report.
string	object_id	M	Unique identifier (ULID) of object detected. If ASM / DMM is not capable of tracking or identifying an object then it can match the reportID
string	task_id	O	Identifier of task that the ASM was carrying out when this detection occurred. By default zero
string	state	O	Whether special case such as object lost
oneof	location_oneof	M	Either Location (see 4.3.1) or RangeBearing (see 4.3.3) must be provided to give the location of the detection
float	detection_confidence	O	Probability that the ASM has detected a real object Value

Type	Field Name	Status	Description
			between 0 and 1. 0 is low confidence, 1 is certain
TrackObjectInfo[]	track_info	O List	Additional metadata about the track, see 0
PredictedLocation	predicted_location	O	Prediction of where the object will be at a later time
TrackObjectInfo[]	object_info	O List	Additional metadata about the detected object, see 0
string	colour	O	Text colour description of the detected object
string	id	O	Unique identifier of this individual object e.g. serial number
DetectionReport Classification[]	classification	O List	List of possible types of object this could be. E.g. Human, Vehicle. Also what the probability of each class is, cf. 6.2.1
DetectionReport Behaviour[]	behaviour	O List	List of possible activities this object could be doing. E.g. Walking, running. Also what the probability of each behaviour is
AssociatedFile[]	associated_file	O List	List of urls to associated media such as image snapshots, 3d point clouds etc.
Signal[]	signal	O List	List of signals detected as part of this detection
AssociatedDetection[]	associated_detection	O List	In a fused detection, the list of detections that were used to generate this detection
DerivedDetection[]	derived_detection	O List	A list of fused detections derived from this detection
oneof	velocity_oneof	O	Velocity of detected object in the specified coordinate system (see 6.2.13 - 6.2.17)

6.2.1 Detection Report - DetectionReportClassification

Type	Field Name	Status	Description
float	confidence	O	Probability that the object is of this object type (class)

SubClass[]	sub_class	O List	List of possible sub classes of object that this object could be (see 6.2.2)
string	type	M	Object type as defined in Appendix A.1 e.g. Human, Vehicle

6.2.2 Detection Report - SubClass

Type	Field Name	Status	Description
string	type	M	Object type as defined in Appendix A.1e.g. Human, Vehicle
float	confidence	O	Probability that the object is of this object type (class)
int32	level	M	Level of the sub class
SubClass[]	sub_class	O	List of possible sub classes of object that this object could be

6.2.3 Detection Report - DetectionReportBehaviour

Type	Field Name	Status	Description
float	confidence	O	Probability that the object is undertaking each activity is
String	type	M	Object activity as defined in Appendix A.2 e.g. walking, running.

6.2.4 Detection Report - Associated File

Type	Field Name	Status	Description
String	type	M	Type of file e.g. image
String	url	M	URL to the media. Typically the sensor identifier will be included as a sub folder in the URL

6.2.5 Detection Report - TrackObjectInfo

Type	Field Name	Status	Description
string	type	M	Name value pairs of metadata associated with track
string	value	M	Value of metadata associated with track/object
float	error	O	Error value associated with value

6.2.6 Detection Report – Predicted Location

Type	Field Name	Status	Description
oneof	predicted_location_oneof	M	Either Location (see 4.3.1) or RangeBearing (see 4.3.3) must be provided to give the location of the predicted detection
google.protobuf.Timestamp	predicted_timestamp	O	Timestamp of predicted detection in UTC if different from Detection Report timestamp

6.2.7 Detection Report - Signal

Type	Field Name	Status	Description
float	amplitude	M	Signal Amplitude – typically in dB Units should be included in Registration – detectionDefinition see 5.5.2
float	start_frequency	O	Minimum Frequency of signal, typically in MHz. Units should be included in Registration – detectionDefinition see 5.5.2
float	centre_frequency	M	Centre Frequency of signal, typically in MHz. Units should be included in Registration – detectionDefinition see 5.5.2
float	stop_frequency	O	Maximum Frequency of signal, typically in MHz. Units should be included in Registration – detectionDefinition see 5.5.2
float	pulse_duration	O	Duration of signal pulse, typically in ns. Units should be included in Registration – detectionDefinition see 5.5.2

6.2.8 Detection Report – Velocity one of

The velocity_oneof field shall be set to the velocity of the detected object in one of the following velocity types:

- Velocity in the Range-rate, Yaw, Pitch frame of reference - This shall use RYPVelocity type (see 6.2.14)
- Velocity in the Ground speed, Heading, Climb frame of reference - This shall use GHCVelocity type (see 6.2.13)
- Velocity in the East, North, Up frame of reference - This shall use ENUVelocity type (see 6.2.15)
- Velocity in the Scalar, Heading, Pitch frame of reference - This shall use SHPVelocity type (see 6.2.16)
- Velocity in the Range, Azimuth, Elevation frame of reference. - This shall use RAEVelocity type (see 6.2.17)

6.2.9 Detection Report - LocationVelocity

Type	Field Name	Status	Description
double	x	O	Velocity in the x axis
double	y	O	Velocity in the y axis
double	z	O	Velocity in the z axis
double	x_error	O	Error in the x-axis value
double	y_error	O	Error in the y-axis value
double	z_error	O	Error in the z-axis value
LocationVelocityUnits enum	units	M	Units the velocity is reported in (see 6.2.10)

6.2.10 LocationVelocityUnits Enumeration

This specifies the units of the velocity being reported. Each enumeration name shall be prefixed with LOCATION_VELOCITY_UNITS_.

0. UNSPECIFIED – location velocity units not defined
1. MS – metres per second
2. KPH – kilometres per hour
3. MPH – miles per hour

6.2.11 Detection Report - RangeBearingVelocity

Type	Field Name	Status	Description
float	elevation	O	Rate of change in elevation
float	azimuth	O	Rate of change in azimuth
float	range	O	Rate of change in range
float	elevation_error	O	Error in the rate of change in elevation
float	azimuth_error	O	Error in the rate of change in azimuth
float	range_error	O	Error in the rate of change in range
RangeBearingVelocityUnits enum	units	M	Units the velocity is reported in (see 6.2.12)

6.2.12 RangeBearingVelocityUnits Enumeration

This specifies the units of the velocity being reported. Each enumeration name shall be prefixed with RANGE_BEARING_VELOCITY_UNITS_.

0. UNSPECIFIED – range bearing velocity units not defined
1. DS – degrees per second
2. RS – radians per second

6.2.13 Detection Report – GHCVelocity

Provide velocity in global coordinates in the style of Air Traffic Management

Type	Field Name	Status	Description
double	ground_speed	M	Speed over the ground
double	heading	O	Azimuth angle of velocity vector
double	climb_rate	O	Rate of change of altitude
double	ground_speed_error	O	Error in the speed over the ground
double	heading_error	O	Error in the azimuth angle of velocity vector
double	climb_rate_error	O	Error in the rate of change of altitude

6.2.14 Detection Report – RYPVelocity

Provide velocity relative to sensor location and sensor pointing direction

Type	Field Name	Status	Description
double	range_rate	O	Rate of change in range
double	yaw_rate	O	Rate of change in yaw
double	pitch_rate	O	Rate of change in pitch
double	range_rate_error	O	Error in the rate of change in range
double	yaw_rate_error	O	Error in the rate of change in yaw
double	pitch_rate_error	O	Error in the rate of change in pitch

6.2.15 Detection Report – ENUVelocity

Provide velocity as a vector in global Cartesian coordinates. This is a revision of the LocationVelocity encoding velocity as per the Location field but with eastings, northings and up for clarity. This aligns with the Location (Cartesian) coordinates system.

Type	Field Name	Status	Description
double	east_rate	M	Velocity in the east-axis (x)
double	north_rate	M	Velocity in the north-axis (y)
double	up_rate	O	Velocity in the up-axis (z)
double	east_rate_error	O	Error in the velocity in the east-axis
double	north_rate_error	O	Error in the velocity in the north-axis
double	up_rate_error	O	Error in the velocity in the up-axis

6.2.16 Detection Report – SHPVelocity

Provide velocity as a vector in global spherical coordinates

Type	Field Name	Status	Description
double	speed	O	Scalar magnitude of velocity vector
double	heading	O	Azimuth angle of velocity vector
double	pitch	O	Pitch (elevation) angle of velocity vector
double	speed_error	O	Error in the scalar magnitude of velocity vector
double	heading_error	O	Error in the azimuth angle of velocity vector
double	pitch_error	O	Error in the pitch (elevation) angle of velocity vector

6.2.17 Detection Report – RAEVelocity

Provide velocity relative to sensor location and ground plane

Type	Field Name	Status	Description
double	range_rate	O	Rate of change in range
double	azimuth_rate	O	Rate of change in azimuth
double	elevation_rate	O	Rate of change in elevation
double	range_rate_error	O	Error in the rate of change in range
double	azimuth_rate_error	O	Error in the rate of change in azimuth
double	elevation_rate_error	O	Error in the rate of change in elevation

6.2.18 Detection Report – associatedDetection

This is a list of the IDs of detection reports with different object IDs related to this Detection Report. For example, for an ASM that can provide detections of more than one type, if they are from the same object at the same time but with different object IDs, they should be linked using this list. If appropriate, the relationship between the detection messages should be included: 'P' for parent, 'C' for child, 'S' for sibling. This may be particularly useful where an ASM can provide many different but related information about an object e.g. when a person is carrying multiple emitting devices.

If used by an DMM this list can be used to provide the list of individual sensor detections that were used to generate a fused detection.

NB Tracks of objects should not be linked like this. Tracks should be a series of detection reports with the same Object ID.

Type	Field Name	Status	Description
google.protobuf.Timestamp	timestamp	O	Timestamp of associated detection
string	node_id	M	Unique identifier of ASM or DMM that was the source of the detection
string	object_id	M	Unique identifier of the detection
AssociationRelation enum	association_type	O	Relationship of associated detection to this detection (see 6.2.19)

6.2.19 AssociationRelation Enumeration

This specifies the relationship to the associated detection. Each enumeration name shall be prefixed with ASSOCIATION_RELATION_.

0. UNSPECIFIED – association relation not defined
1. NO_RELATION – there is no relation to this detection
2. PARENT – the associated detection is a parent of this detection
3. CHILD – the associated detection is a child of this detection
4. SIBLING – the associated detection is a sibling of this detection

6.2.20 Detection Report – derivedDetection

This is used by an DMM. When a fused detection is generated, this field can be used to add a reference to the fused detection from the record of the original detection. Used with associatedDetection, this provides two linked lists linking original and derived, fused detections.

Type	Field Name	Status	Description
google.protobuf.Timestamp	timestamp	O	Timestamp of associated detection
string	node_id	M	Unique identifier of ASM or DMM that was the source of the detection
string	object_id	M	Unique identifier of the detection

6.3 Alert Message

The primary use of the Alert message is for the DMM to provide Alerts to the Operator via the UI. It can also be used by a node to inform the DMM that it is about to do something or that something unusual has happened. An example of this might be a sensor deciding to do a detailed scan of an object or area.

For information only alert messages, where no response is expected, place ‘Information’ in the alertType field. If the alert message merits a response from the DMM, the Alert Response message can be used.

This description field within the Alert message can also be used to pass auxiliary information on the status of the node directly to the UI.

Type	Field Name	Status	Description
string	alert_id	M	Unique identifier of alert (ULID)
AlertType enum	alert_type	O	Type of alert. 'Information' implies no response required. See 6.3.1
AlertStatus enum	status	O	Current status of alert: Active, Acknowledge, Reject, Ignore, Clear (see 6.3.2)
string	description	O	Text Description of Alert
LocationOrRangeBearing	location	O*	Providing a Cartesian location is optional and should only be done if relevant to do so
string	region_id	O	Optional field to refer to an existing Region of Interest as designated by the DMM (ULID)
DiscretePriority enum	priority	O	'Low', 'Medium' 'High' (see 6.3.3)
float	ranking	O	0.0 – 1.0
float	confidence	O	0.0 – 1.0
string	additional_information	O	Any additional information for the DMM
AssociatedFile[]	associated_file	O List	List of associated data files. See 6.2.4
AssociatedDetection[]	associated_detection	O List	List of associated detection points. See 6.2.13

6.3.1 AlertType Enumeration

This specifies the severity of alert being reported. Each enumeration name will be prefixed with ALERT_TYPE_.

0. UNSPECIFIED – alert type is undefined
1. INFORMATION – this alert is for information only

2. WARNING – this alert shows something that may need a human to review it
3. CRITICAL – this node has a serious problem and may fail soon
4. ERROR – the node has a problem but is still operating
5. FATAL – the node has failed
6. MODE_CHANGE – a node wishes notify the DMM that it wishes to autonomously change mode

6.3.2 AlertStatus Enumeration

This specifies the current state of the alert being reported. Each enumeration name shall be prefixed with ALERT_STATUS_.

0. UNSPECIFIED – alert status is not defined
1. ACTIVE – alert is active and has yet to be responded to
2. ACKNOWLEDGE – alert has been previously acknowledged
3. REJECT – alert has been previously rejected
4. IGNORE – alert has been previously ignored
5. CLEAR – enables a previously acknowledged alert to be ‘cleared’

6.3.3 DiscretePriority Enumeration

This specifies the priority in discrete steps. It is normally used to define priorities. Each enumeration name shall be prefixed with DISCRETE_PRIORITY.

0. UNSPECIFIED – discrete priority not defined
1. LOW – low priority set
2. MEDIUM – medium priority set
3. HIGH – high priority set

6.3.4 Alert - Associated File

Type	Field Name	Status	Description
string	type	M	Type of file e.g. image
string	url	M	URL to the media. Typically the sensor identifier will be included as a sub folder in the URL

6.4 Alert Acknowledgement Message

These short messages complete the handshaking, and allow a DMM to respond to an alert. The ‘destination_id’ field described at the start of section 6 shall be populated when an Alert Acknowledgement message is generated.

Type	Field Name	Status	Description
string	alert_id	M	Alert ID the acknowledgement is in response to
AlertStatus enum	alert_status	M	The node’s response to the tasking request (see 6.4.1)
string	reason	O	The reason the task was rejected if the task was rejected

6.4.1 AlertStatus Enumeration

This specifies the status of the alert. Each enumeration name shall be prefixed with ALERT_STATUS_.

0. UNSPECIFIED – the alert status is not defined
1. ACCEPTED – the requested alert has been accepted
2. REJECTED – the requested alert has been rejected
3. CANCELLED – the requested alert has already been cancelled

6.5 Task Message

These are messages sent by the DMM to an individual node, commanding it to undertake a specific task. The overall philosophy used is that commands and filtering options are task-based with an optional associated region defined in the task command.

A node must acknowledge a Task.

If an node cannot comply with a control message, it sends a TaskACK acknowledgement message with an appropriate rejection message.

The ‘destination_id’ field described at the start of section 6 shall be populated when a Task message is sent.

Type	Field Name	Status	Description
string	task_id	M	See 6.5.1
string	task_name	O	Optional name of task
string	task_description	O	Optional description of task
google.protobuf.Timestamp	task_start_time	O	UTC time to start task
google.protobuf.Timestamp	task_end_time	O	UTC time to end task

Control enum	control	M	‘Start’,’Stop’, ‘Pause’,’Default’ - See 6.5.2
Region[]	region	O List	Optional List of regions associated with the task. See 6.5.3
Command	command	O	Populated if this task is a command. See 6.5.4

6.5.1 Task - Task ID

Task messages shall include a task ID. The task ID is a ULID generated by the DMM to uniquely identify a task so that the acknowledgement message and any subsequent Detection Report messages have a reference. Task ID which is blank indicates the default task to be undertaken when there is no other task allocated to the node.

6.5.2 Control Enumeration

This field indicates what to do with this task. Each enumeration name shall be prefixed with CONTROL_.

0. UNSPECIFIED – no control defined

1. START - Initialises and schedules or starts the task or restarts a paused task.

2. STOP - stop the task, remove its definition, revert to previous task.

3. PAUSE - stop the task but keep it defined for later restart, revert to previous task.

4. DEFAULT – (without other fields populated in the message) - start or revert to the default task, stopping all other tasks. Or (when other fields populated in the message) - defines the default task to be undertaken when no other tasks are defined.

   NB. A stop message does not require a ‘region’ or ‘command’ section.

6.5.3 Task - Region

Region tasks are designed to control the geographical area within the coverage of steerable nodes or field of view of fixed nodes that the nodes should report detections from. Typically by default, the node will detect from its full field of view and so a region task will limit the area to report from.

In addition, if supported by the node, object type can be used to restrict the types of objects that will be reported.

Any task overrides the default task. If a second override task comes and the node can't support multiple tasks then it needs to report that in the acknowledgement message when tasked with the second task. If regions are to be removed from the default task then a default task message is sent with all remaining regions.

Type	Field Name	Status	Description
RegionType enum	type	M	Area of Interest, Ignore, Boundary.
string	region_id	M	Region ID Number generated by the DMM
string	region_name	M	e.g. Car Park
LocationOrRangeBearing	region_area	M	This should either be reported as a 'RangeBearingCone' proto message (see 4.3.4) or a 'LocationList' proto message (see 4.3.2). For a location list, points should be in order around the boundary of the region.
ClassFilter[]	class_filter	O List	List of class Filters to apply to region. If omitted then no filtering is applied.
BehaviourFilter[]	behaviour_filter	O List	List of behaviour Filters to apply to region. If omitted then no filtering is applied.

6.5.3.1 Task – Region - Class Filter

Type	Field Name	Status	Description
parameter	parameter	M	Typically filter on confidence value
subClassFilter[]	sub_class_filter	O List	List of sub classes to filter on
string	type	M	Class to filter on
DiscreteThreshold enum	priority	O	Priority of this filter

6.5.3.2 DiscreteThreshold Enumeration

This specifies the threshold in discrete steps. It is normally used to define priorities. Each enumeration name shall be prefixed with DISCRETE_THERSHOLD_.

0. UNSPECIFIED – discrete threshold not defined
1. LOW – low threshold set
2. MEDIUM – medium threshold set
3. HIGH – high threshold set

6.5.3.3 Task – Region - SubClass Filter

Type	Field Name	Status	Description
Parameter	parameter	M	Typically filter on confidence value
string	type	M	Class to filter on
SubClassFilter[]	sub_class_filter	O	List of any child sub classes to filter on
DiscreteThreshold enum	priority	O	Priority of this filter (see 6.5.3.2)

6.5.3.4 Task – Region - Behaviour Filter

Type	Field Name	Status	Description
parameter	parameter	M	Typically filter on confidence value
string	type	O	Behaviour to filter on
DiscreteThreshold enum	priority	O	Priorirty of this filter (see 6.5.3.2)

6.5.4 Task – Command

This section of the message is populated for command based tasks. These tend to be short lived or instantaneous whereas Region ones tend to persist until another task is received. Only one command message should be issued at a time and so only one of the top level fields should be populated.

Type	Field Name	Status	Description
oneof	command	M	See 6.5.4.1 - 6.5.4.6
string	command_parameter	O	Optional parameter string for some commands

6.5.4.1 Request Commands

The Request command with a parameter of “Registration” allows the DMM to request the node to send the initialisation message again.

The Request command with a parameter of “Heartbeat” allows the DMM to request a full Status Report message with the current value of all fields supported by that node.

The Request command with a parameter of “Reset” allows the DMM to stop the node’s current task and return to its default state

The Request command with a parameter of “Stop” allows the DMM to request that the node stop sending all Detection Report messages. The node should continue sending Status Report messages to maintain the connection to the data agent.

The Request command with a parameter of “Start” makes the ASM resume sending Detection Report messages after a Request – Stop command.

6.5.4.2 Detection Threshold Commands

The individual ASMs are expected to exhibit a level of autonomy. Therefore the DMM is not expected to use a “long screwdriver” to set low-level parameters within an individual ASM. The DMM may however choose to request an ASM to adjust its internal thresholds to reduce its FAR, or increase its P(d), based for example on comparing the detection performance of one ASM with another with an overlapping field of view.

This is covered by the command ‘DetectionThreshold’. This allows the DMM to adjust the ASMs detection threshold to ‘Low’, ‘Medium’ or ‘High’ or set it ‘Lower’ or ‘Higher’ or return it to its own ‘Auto’ threshold.

6.5.4.3 Detection Report Rate Commands

The DMM may wish to request an individual ASM to alter the frequency at which it reports data, for example to throttle back the detections if they were supplied too frequently. This is covered by the command ‘DetectionReportRate’ which can be changed up or down by the DMM, or used to request a specific output rate. This allows the DMM to adjust the ASMs report rate to ‘Low’, ‘Medium’ or ‘High’ or set it ‘Lower’ or ‘Higher’ or return it to its own ‘Auto’ rate.

6.5.4.4 Classification Threshold Commands

The DMM may wish to request an individual ASM to alter the sensitivity at which it classifies data, for example to allow for ‘looser’ searches in challenging environments. This is covered by the command ‘classificationThreshold’ which can be changed up or down by the DMM, or used to request a classification threshold. This allows the DMM to adjust the ASMs classification threshold to ‘Low’, ‘Medium’ or ‘High’ or set it ‘Lower’ or ‘Higher’ or return it to its own ‘Auto’ threshold.

6.5.4.5 Mode Commands

The ‘mode’ message forces the sensor to change to one of its operating modes defined in the Initialisation message. If a sensor only supports one mode, then this command can be ignored.

6.5.4.6 Look At Commands

The ‘lookAt’ command tasks the ASM to suspend its current tasking, take a quick look at a specific location or range bearing (see 4.3.5) to see if it can see anything and then return to its on-going tasking. It will return a detection message if it detects an object at that location, or a ‘SensorTaskACK’ Acknowledgement message with the ‘Status’ field “Nothing found”. Both will include the task ID of the ‘lookAt’ command.

The ASM will determine the most appropriate field of view to use in response to the ‘lookAt’ command. If this is not the default field of view, then this will be reported by a heartbeat message to accompany any detection message response to the ’lookAt’ command.

6.6 Task Acknowledgement Message

These short messages complete the handshaking, and allow an ASM to accept or reject a task. If an ASM cannot comply with a control message, it sends a TaskACK acknowledgement

message with an appropriate error message. The ‘destination_id’ field described at the start of section 6 shall be populated when the Task Acknowledgement message is sent.

Type	Field Name	Status	Description
string	task_id	M	Task ID the acknowledgement is in response to
TaskStatus enum	task_status	M	The node’s response to the tasking request (see 6.6.1)
string	reason	O	The reason the task was rejected if the task was rejected, or failed
AssociatedFile	associated_file	O	File associated with task being acknowledged

6.6.1 TaskStatus Enumeration

This specifies the status of the task. Each enumeration name shall be prefixed with TASK_STATUS_.

0. UNSPECIFIED – the task status has not been defined
1. ACCEPTED – the requested task has been accepted and will be carried out
2. REJECTED – the requested task has been rejected by the ASM
3. COMPLETED – the requested task has already been completed
4. FAILED – the node has accepted the task and has tried to complete the task but has not succeeded in doing so.

6.7 Error Messages

If an invalid message is received by the Middleware from either an ASM or DMM client then the Middleware will respond by sending an error message back to the source. This contains the time of the message the content of the invalid message and a message describing the error. The ‘destination_id’ field described at the start of section 6 shall be populated when an Error message is sent.

Type	Field Name	Status	Description
bytes	packet	M	The packet or part of the message which caused the error
string	error_message	M	Description of the error being reported

7 Example Messages

This section contains examples of common messages. They show a camera-based ASM registering, sending a status report and detection report. The ASM is then tasked to complete a Look At command by the DMM. Note that the messages here are serialised into JSON. In a production environment, the messages should be serialised into binary.

7.1 Example Registration Message


```
{
  "timestamp": "2022-11-03T10:05:06.141204500Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "registration": {
    "nodeType": [
      "CAMERA"
    ],
    "name": "MyCamera ASM",
    "capabilities": [
      {
        "category": "Test",
        "type": "Camera",
        "units": "Url"
      }
    ],
    "heartbeatDefinition": {
      "heartbeatInterval": {
        "units": "SECONDS",
        "value": 5
      },
      "locationDefinition": {
        "locationUnits": "LATLNGDEGM",
        "locationDatum": "WGS84E"
      },
      "heartbeatReport": [
        {
          "category": "SENSOR",
          "type": "SensorLocation",
          "onChange": true
        },
        {
          "category": "POWER",
          "type": "status",
          "units": "OK, Fault",
          "onChange": true
        }
      ]
    },
    "modeDefinition": [
      {
        "modeName": "Default",
        "modeType": "Permanent",
        "settleTime": {
```

```
      "units": "SECONDS",
      "value": 1
    },
    "scanType": "Steerable",
    "trackingType": "Tracklet",
    "detectionDefinition": {
      "locationType": {
        "locationUnits": "LATLNGDEGM",
        "locationDatum": "WGS84E"
      },
      "detectionPerformance": [
        {
          "type": "FAR",
          "units": "Per Second",
          "unitValue": "1",
          "variationType": "Linear with range"
        }
      ],
      "detectionClassDefinition": [
        {
          "confidenceDefinition": "SINGLE_CLASS",
          "classDefinition": [
            {
              "type": "Human",
              "units": "probability",
              "subClass": [
                {
                  "type": "Male",
                  "units": "probability",
                  "level": 1
                },
                {
                  "type": "Female",
                  "units": "probability",
                  "level": 1
                }
              ]
            }
          ]
        }
      ]
    },
    "task": [
      {
        "regionDefinition": {
          "regionType": [
            "IGNORE"
          ],
          "settleTime": {
            "units": "SECONDS",
            "value": 1
          },
          "regionArea": [
```

```
                {
                  "locationUnits": "LATLNGDEGM"
                }
              ]
            },
            "command": [
              {
                "name": "REQUEST",
                "units": "Location",
                "completionTime": {
                  "units": "SECONDS",
                  "value": 1
                }
              }
            ],
            "type": "REQUEST"
          }
        ]
      }
    ]
  }
}
```


7.2 Registration Ack Message


```
{
  "timestamp": "2022-11-03T10:10:32.248604200Z",
  "nodeId": "139bf8c7-84da-4c00-a068-4d58dca747b8",
  "registrationAck": {
    "destinationId": "b902eb31-97f1-4acf-b994-da3ec901074e"
  }
}
```


7.3 Status Report Message


```
{
  "timestamp": "2022-11-03T10:07:24.003509600Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "statusReport": {
    "reportId": "01GGYFBAV4EEPB1398MKQ47E6E",
    "system": "OK",
    "power": {
      "source": "mains",
      "status": "Ok",
      "level": 1
    },
    "sensorLocation": {
      "x": 51.1739726374,
      "y": -1.82237671048,
      "z": 788,
      "coordinateSystem": "LATLNGDEGM",
      "datum": "WGS84E"
    }
```

```
  }
}
```

7.4 Detection Report Message

```
{
  "timestamp": "2022-11-03T10:07:24.079937400Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "detectionReport": {
    "reportId": "01GGYFBAXGDG7AGAHRZ6XSNY12",
    "objectId": "01GGYFBAXH4VYRQYEX7S3XGK3H",
    "taskId": "01GGYFBAXHNV9DN0N74DFX2952",
    "location": {
      "x": 51.1739726374,
      "y": -1.82237671048,
      "z": 828
    },
    "detectionConfidence": 0.99,
    "classification": [
      {
        "type": "Air Vehicle",
        "confidence": 1
      }
    ]
  }
}
```

7.5 Task Message

```
{
  "timestamp": "2022-11-03T10:29:39.941877200Z",
  "nodeId": "139bf8c7-84da-4c00-a068-4d58dca747b8",
  "task": {
    "destinationId": "b902eb31-97f1-4acf-b994-da3ec901074e",
    "taskId": "01GGYGM3F6FHSDDMVCQJ11ZNBF",
    "control": "START",
    "command": {
      "lookAt": {
        "locationList": {
          "locations": [
            {
              "x": 51.1739726,
              "y": -1.8223767,
              "z": 790,
              "coordinateSystem": "LATLNGDEGM",
              "datum": "WGS84E"
            }
          ]
        }
      }
    }
  }
}
```

7.6 Task Ack Message


```
{
  "timestamp": "2022-11-03T10:33:04.819389200Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "taskAck": {
    "destinationId": "139bf8c7-84da-4c00-a068-4d58dca747b8",
    "taskId": "01GGYGM3F6FHSDDMVCQJ11ZNBF"
  }
}
```

A Classification Taxonomy

The successful operation of a system relies on defining a taxonomy of object classes, behaviours and other characteristics. The taxonomy defines a set of descriptive nouns in a hierarchy with three levels. An ASM should provide the lowest level of classification that is within its capabilities.

A.1 Object Classification Taxonomy

Table C1 below contains the full list of class names in the SAPIENT Object Classification Taxonomy. The Unknown class is reserved for the situation where the ASM cannot place an object in any of the other classes. It means 'I am certain this is an unknown class' rather than 'I can't tell what class this is'.

Class	Sub Class Level 1	Sub Class Level 2
Unknown		
Human	Male	
	Female	
Animal	Bird	
	Horse	
	Other	
Fixed Object	Building	
	Fixed Tower/mast	
	Road	
	Bridge	
	Dam	
	Street Furniture	
	Natural feature	
	Other	
EM Transmitter	Wifi AP	
	Broadcast	
	UAS Control Station	
	UAV Transmitter	
	Radar	
	RF Jammer	
	Laser/Lidar	
	RF DEW/EM Pulse	
	Other	
Equipment	Weapon	Gun
		Missile Launcher
		Missile
		Bomb
		IED/mine

Class	Sub Class Level 1	Sub Class Level 2
		Grenade
		Other
	Passive Sensor	Camera
		Acoustic
		Other
	Other	
Land Vehicle	Tracked	Commercial
		Military
	>2 wheels - Heavy	Commercial
		Military
	>2 wheels - Medium	Commercial
		Military
	>2 wheels - Light	Commercial
		Military
	2 wheels	Motorbike
		Bicycle
Sea Vessel	Large Ship	Commercial
		Military
	Small boat	Commercial
		Military
	Submarine	Commercial
		Military
Air Vehicle	Manned Rotary Wing	Commercial
		Military
	Manned Fixed Wing	Commercial
		Military
	UAV Rotary Wing	Commercial
		Military
	UAV Fixed Wing	Commercial
		Military

Table C1: SAPIENT Object Classification Taxonomy

A.2 Object Behaviour Taxonomy

Where object behaviour can be detected, this is reported independently of object type using the behaviour types. Any confidence values associated with the behaviour

should be expected to relate to the object class type with the highest confidence. E.g. If an object is detected as person with a probability of 0.6 and animal with a probability of 0.3, any behaviour probability is expected to be in relation to it being a person.

The following is the current core list of behaviours. This list can be easily extended in future releases of the ICD. Not all of these behaviours will necessarily be supported on a given system. This is likely to be implementation specific.

- Walking
- Running
- Crawling
- Climbing
- Digging
- Throwing
- Loitering
- Active
- Passive

Last page