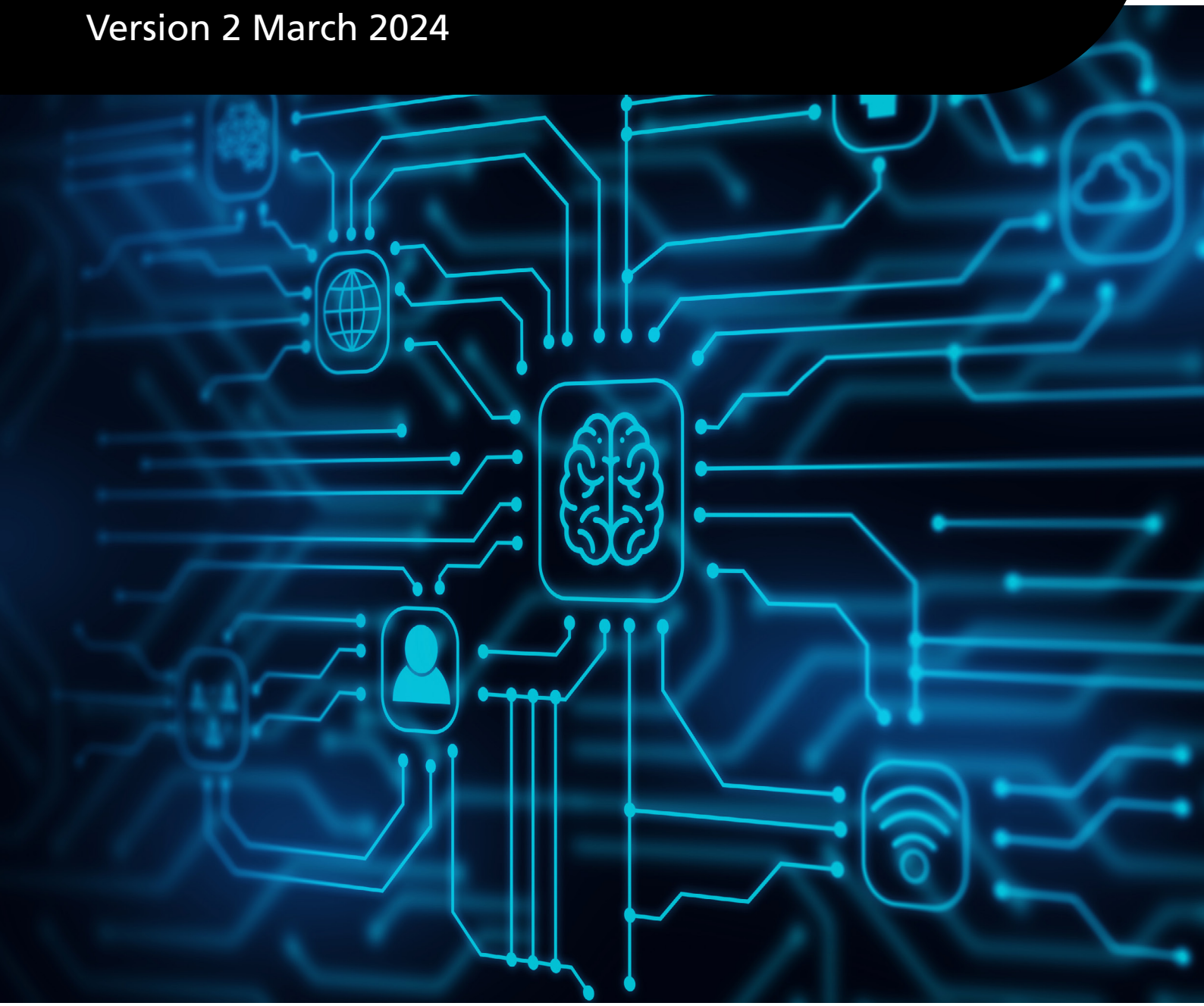
SAPIENT Network of autonomous sensors and effectors – Interface control document – Specification

Version 2 March 2024



BSI Flex 335 v2.0:2024-03

[dstl]
The Science Inside

SAPIENT

bsi

Publishing and copyright information

The BSI copyright notice displayed in this document indicates when the document was last issued.

Published by BSI Standards Limited 2024

ISBN 978 0 539 26777 8

ICS 35.030, 35.080, 95.040

Release History
First Version July 2023
Second Version March 2024

Contents

Annexes

List of figures

List of tables

Foreword

This BSI Flex was sponsored by the UK Ministry of Defence. Its development was facilitated by BSI Standards Limited and it was released under licence from The British Standards Institution. It came into effect on 31 March 2024.

Acknowledgement is given to Paul Thomas, DSTL, as the lead technical author, and the following organizations that were involved in the development of this BSI Flex as members of the Advisory Group:

- Createc
- Defence UAS CDC
- Defence Science and Technology Laboratory (DSTL)
- Exensor
- L3Harris
- Leonardo
- Marss
- MBDA
- Metis Aerospace
- OSL
- Overview
- Plextek
- QinetiQ
- Roke
- Thales UK

This BSI Flex is not to be regarded as a PAS or British Standard.

The BSI Flex process enables a standard to be rapidly developed, on an iterative basis, in order to fulfil an immediate stakeholder need. A BSI Flex can be considered for further development as a PAS or British Standard, or constitute part of the UK input into the development of a European or international standard.

The content in this version is part of an iterative process. It is likely to change from time to time with subsequent iterations.

Relationship with other publications

This BSI Flex is based on the SAPIENT specification published by Dstl on behalf of the Ministry of Defence, which has subsequently been withdrawn in favour of BSI Flex 335 v1.0.

It provides definitions and guidance for building components compatible with the SAPIENT system, which uses AI and autonomy in networked multi-sensor systems.

Information about this document

This is Version 2 of BSI Flex 335, which has been released to enable stakeholders to engage with the content and feed back comments for further versions of the document to be developed. This is the second public consultation of this BSI Flex. Users are encouraged to comment on this version.

This publication can be withdrawn, revised, partially superseded or superseded. Information regarding the status of this publication can be found in the Standards Catalogue on the BSI website at bsigroup.com/standards, or by contacting the Customer Services team.

Where websites and webpages have been cited, they are provided for ease of reference and are correct at the time of publication. The location of a webpage or website, or its contents, cannot be guaranteed.

Presentational conventions

The provisions of this document are presented in roman (i.e. upright) type. Its requirements are expressed in sentences in which the principal auxiliary verb is "shall".

Commentary, explanation and general informative material is presented in smaller italic type, and does not constitute a normative element.

Where words have alternative spellings, the preferred spelling of the *Shorter Oxford English Dictionary* is used (e.g. "organization" rather than "organisation").

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

Compliance with a BSI Flex cannot confer immunity from legal obligations.

0 Introduction

0.1 General

This BSI Flex 335 v2.0 is based on BSI Flex 335 version 1.0, which was based on the Sensing for Asset Protection with Integrated Electronic Networked Technology (SAPIENT) Interface Control Document (ICD) v7.0 that was published on the gov.uk website by the Defence Science and Technology Laboratory.

Terminology transitioned between version 7.0 of the SAPIENT ICD and BSI Flex 335 v1.0. This document assumes familiarity with the new terminology. Any reader unfamiliar with this transition may find guidance in BSI Flex 335 v1.0.

Major changes for this version include:

a) addressing use cases involving groupings of edge nodes on a single platform (either mobile or pointable), where platform movements affect all edge nodes (see **4.10**);

b) addressing use cases involving groupings of edge nodes deployed as a hierarchy of fusion nodes (either due to aggregation of fusion nodes that cover local regions, or for fusion nodes that control the edge nodes on a structure) (see **4.11**);

c) adding support for dynamically extensible taxonomy, to allow edge nodes to declare the taxonomy extensions they will use at registration time and then use them at detection time;

d) adding extensions to tasking commands to support mobile nodes (including Move_To, Patrol and Follow) (**Table 58** and **Table 59**);

e) minor changes to some field status descriptors; and

f) general tidy up of clause titles to better reflect the .proto files that accompany this BSI Flex (see **5.1**).

A full list of changes is in **Annex A**.

This BSI Flex is accompanied by .proto files which are the machine-readable version of the standard. Additional information about the .proto files is available in **C.2**. To guarantee correct operation between SAPIENT components, it is advisable for all components to use the same .proto files. A developer who generates their own .proto files for their component cannot be certain their component will work with other SAPIENT components using different .proto files.

Some useful websites are listed in the Bibliography.

0.2 SAPIENT concept

SAPIENT is a concept that describes a system comprised of a suite of autonomous edge nodes, which may have sensor or effector (or both) capabilities combined with remote modules that perform fusion and node tasking.

The key principles of SAPIENT are:

a) local autonomy at the edge, i.e. local processing of sensor data where the nodes are sensor edge nodes;

b) the transmission of information (the output of the processing) rather than the raw data from the sensor edge node;

c) the fusion of information at remote fusion nodes and the generation of tasking messages for taskable edge nodes (sensor edge nodes or effector nodes); and

d) local autonomous execution of received taskings at taskable edge nodes (sensor edge nodes or effector nodes).

When combined as a system, these principles give the following desirable features:

1) reduction in operator workload during monitoring activities;

2) improved autonomy for decision making;

3) improved autonomy for management of edge nodes;

4) lower bandwidth requirements for network traffic; and

5) node modularity – nodes of different types (particularly sensor edge nodes) use the same interface for communication whatever their modality.

A typical configuration of a SAPIENT system is described in this document as a suite of edge nodes feeding information into, and being tasked by, a fusion node.

0.3 Applications of SAPIENT

The SAPIENT concept is applicable to any situation where a suite of autonomous distributed edge nodes collaborate to build situational awareness. Use cases include asset protection, security, defence, law enforcement, counter-threat, safety, hazard management and environmental monitoring.

SAPIENT is intended to work at the "information level" rather than continually streaming raw data. Each node communicates with recipient nodes via standard messages comprising either declarations and detections, including as much information as the nodes can provide, or taskings, providing instructions for the edge nodes.

The SAPIENT concept shows benefits in situations where no single sensor has all the information required for situational awareness so fusion of the information from multiple sensors is required. Additionally, since SAPIENT explicitly provides for node management, the concept shows increased benefit where nodes can be tasked, i.e. they have degrees of freedom (e.g. look direction, field of view, scan rate, mode of effect). Finally, since SAPIENT explicitly enables bandwidth reduction in the network, it is beneficial where there are bandwidth constraints in the information transmission network. Often this applies in spatially distributed sensing networks.

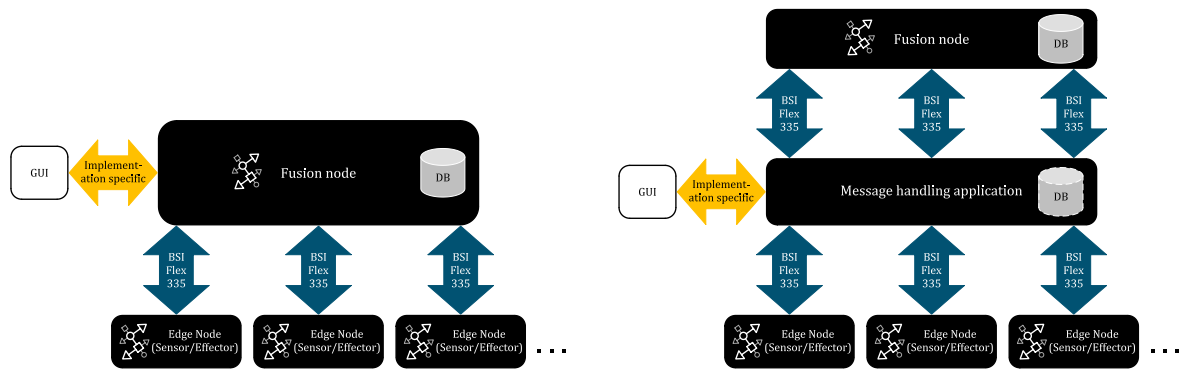
Whilst the SAPIENT philosophy is that fusion functionality is based on the "summary" messages produced by the sensor edge nodes, it is recognized that some use cases require transmission of raw data from sensor edge nodes, e.g. for operator confirmation purposes. This is not part of the SAPIENT concept, but the functionality to facilitate this has been provided in some SAPIENT systems (see **C.4**).

0.4 SAPIENT architecture

The SAPIENT concept adopts a system of systems approach, consisting of multiple edge nodes connected to a fusion node.

There are two configurations for SAPIENT architecture, one where message handling functions are integrated with the fusion node (see **Figure 1**, left-hand side), and one where the message handling applications are separated into a distinct software layer (often called a middleware) (see **Figure 1**, right-hand side). Either choice is valid.

Figure 1 – SAPIENT architecture



***NOTE** This diagram is illustrative only and does not form part of the BSI Flex.*

In both cases, the fusion node acts as the "server", whilst the edge nodes (and the message handling application, if present) acts as the "client", in a client/server architecture.

***NOTE 1** The communication between the fusion node (or message handling application) and any Graphical User Interface (GUI) is implementation specific and not covered by this BSI Flex standard. The aspiration for a future version of BSI Flex 335 is to include an Application Programming Interface (API) to provide a standard mechanism for external systems (which could include a GUI, or other systems) to connect to the core SAPIENT system.*

When the message handling application is separated from the fusion node, as illustrated on the right-hand side of **Figure 1**, it is 'transparent', i.e. it has no effect on the content of the messages or the behaviour of the nodes.

***NOTE 2** The 'transparent' nature of the message handling application is a change from BSI Flex 335 v.1.0 to enable modularity, i.e. the ability to replace nodes (including fusion nodes) in a SAPIENT system.*

For clarity, transparency of the message handling application results in the following:

a) the fusion node is responsible for generating "registration acknowledgement" messages - these are sent by the fusion node upon successful registration of edge nodes.

b) The fusion node is responsible for sending "alert acknowledgement" messages – these are generated by the GUI or fusion node (as appropriate) and sent via the fusion node (and then onwards via the message handling application, if present).

c) There is a requirement for the fusion node to maintain persistent knowledge (as appropriate to the fusion node's function) of registered nodes and their capabilities.

***NOTE 3** Some implementations of message handling applications may choose to maintain their own store of messages and events. However, the fusion node needs to have a local store in case of disconnection.*

***NOTE 4** See **C.3** for more detail on functionality and behaviour of message handling applications.*

The SAPIENT concept is intended to support a range of different sensor and effector modalities, a range of levels of object detection, localization and classification, and a range of interpretations of what "autonomy" means. A standard Interface Control Document (ICD) for SAPIENT therefore strikes a balance between flexibility to accommodate a range of inputs and specifications to achieve a common interface. This is achieved using a mix of mandatory, discretionary and conditional fields, which provide a flexible and extensible interface that can cope with future edge node and Fusion modules in future instantiations of SAPIENT systems.

The message exchange behaviour is illustrated in **Figure 2**, which shows the following sequential activities:

a) registration – the edge nodes register themselves on the system;

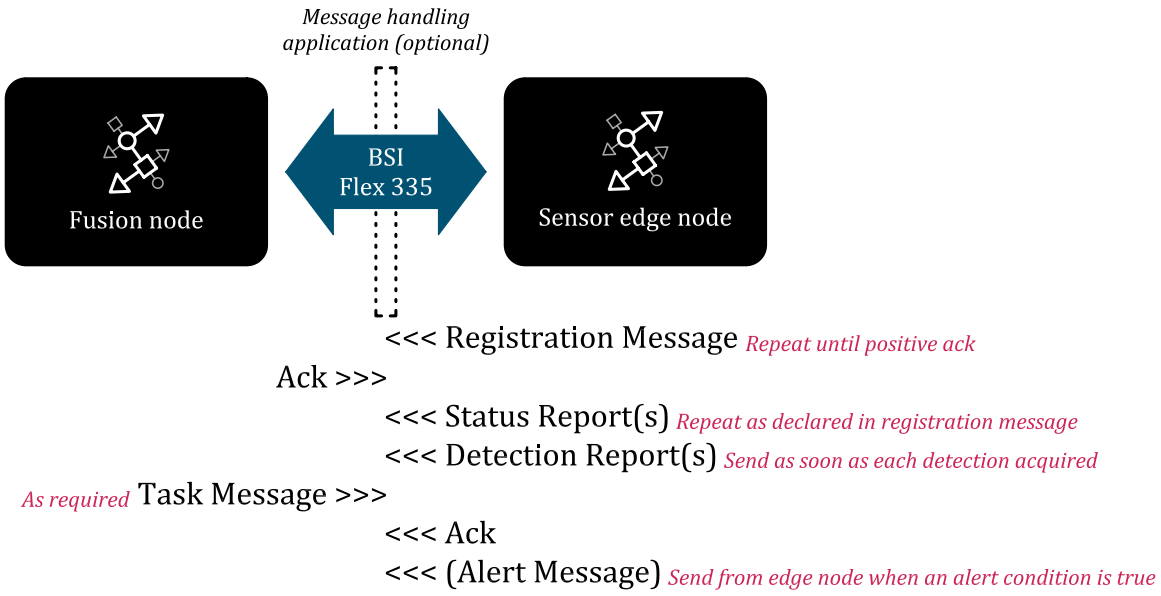
b) tasks – taskable nodes perform tasks; either sensing tasks, if they are sensor edge nodes, or effector tasks if they are effector nodes (edge nodes can in some cases act as both sensor and effector nodes, in which case they perform both type of tasks);

c) detection – sensor edge nodes then send messages upon receiving positive detections;

Taskable edge nodes are nodes that can be tasked with specific sensor or effector objectives from the fusion node.

More detail on normal operation is provided in **4.5**.

Figure 2 – Typical SAPIENT message sequence



The fusion node performs higher-level fusion on the edge node messages. The term fusion describes a spectrum of functions from geospatial aggregation through to correlation, confirmation, de-confliction and task optimization. The output of this processing is information on entities, tracks and alerts (which may be displayed on a Graphical User Interface (GUI)) and tasking of nodes to perform subsequent actions (which may be for sensor edge nodes to provide further information such as imagery of objects of interest or for effector nodes to perform an effector action).

Some message handling functions are required. These can either be carried out by the fusion node (Left-hand side of **Figure 1**) or a specific message handling application (Right-hand side of **Figure 1**) (see **C.3**).

1 Scope

This BSI Flex specifies the interfaces within a SAPIENT (Sensing for Asset Protection with Integrated Electronic Networked Technology) system. Specifically, it addresses the interfaces between edge nodes, the fusion node and any message handling application.

This BSI Flex provides the structure, format and content of the SAPIENT messages and is intended for use by developers of edge nodes and fusion modules. In addition, an annex to this document provides guidance on how a Graphical User Interface (GUI) links to the output of any selected system.

This BSI Flex does not specify specific sensors or effectors, neither does it specify implementation-specific parameters such as security solutions.

2 Normative references

There are no normative references in this document.

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

3.1.1 alert

type of SAPIENT message used for inter-node communication (when an alert condition is true) or node-user communication (for non-detection information)

3.1.2 command-based task

short-lived or instantaneous task message

3.1.3 conditional

SAPIENT field status that indicates under certain conditions the field is to be completed

NOTE *If those conditions are not true the field is to be considered discretionary.*

3.1.4 detection

type of SAPIENT message used to indicate presence of target activity or behaviour

3.1.5 discretionary

SAPIENT field status that indicates the field may or may not be completed

3.1.6 effector node

SAPIENT node that makes an effect on objects in the environment

NOTE *Some sensor edge nodes (**3.1.15** sensor edge node) can also act as effectors.*

3.1.7 fusion node

autonomous SAPIENT node that performs fusion and node tasking functions

3.1.8 mandatory

SAPIENT field status that indicates the field needs to be completed; it is not valid for the field to be left uncompleted

3.1.9 message handling application

optional application that resides on the network and provides message acknowledgement, message routing and information storage/retrieval functions

3.1.10 node

self-contained SAPIENT module, conforming to the SAPIENT architecture

3.1.11 protobuf field rule

rule specifying the information in the protobuf field which can be one of:

a) singular;

b) optional;

c) repeated; or

d) map

NOTE 1 *For further information on the protobuf field see https://developers.google.com/protocol-buffers.*

NOTE 2 *This list is from Protobuf version 3.*

NOTE 3 *These field rules are different from the SAPIENT field status (**3.1.14 SAPIENT field status**)*

3.1.12 region

geographical area defined by a boundary polygon

3.1.13 region-based task

task message based on a region, which persists until another task is received

3.1.14 SAPIENT field status

status specifying the information in the SAPIENT field which can be:

a) mandatory;

b) discretionary; or

c) conditional

NOTE *These field statuses are different from the protobuf field rules (**3.1.11** protobuf field rule)*

3.1.15 sensor edge node

autonomous SAPIENT edge node that performs a sensor function

3.1.16 status

type of SAPIENT message sent by a node (even in the absence of any detections) to indicate its operational status

3.1.17 task

type of SAPIENT message containing an imperative

NOTE *Tasks are sent by the fusion node as an instruction to individual edge nodes.*

3.2 Abbreviated terms

BNC	Bayonet Neill-Concelman
DEW	Directed Energy Weapon
DVI	Digital Visual Interface
EM	Electromagnetic
ESM	Electronic Support Measures
FAR	False Alarm Rate
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
IED	Improvised Explosive Device
MIME	Multipurpose Internet Mail Extensions
NTP	Network Time Protocol
PD	Probability of Detection
PTZ	Pan, Tilt, Zoom (usually refers to a camera system)
RF	Radio Frequency
SAPIENT	Sensing for Asset Protection using Integrated Electronic Networked Technology
SDI	High-Definition Serial Digital Interface
UAS	Uncrewed Aerial System
UAV	Uncrewed Aerial Vehicle
ULID	Universally unique Lexicographically sortable Identifier
URL	Uniform Resource Locator
UTM	Universal Tranverse Mercator
UUID	Universally Unique Identifier

4 Interface description

*COMMENTARY ON CLAUSE **4***

*This clause specifies the interface description in the conventional sequence of operation. Where the term message handling application is used in this clause, this might mean an integrated message handling and fusion node, (see **0.4**). The field status of the SAPIENT fields described in this document are labelled mandatory (**3.1.8**), discretionary (**3.1.5**) and conditional (**3.1.3**). In the tables in this clause, these are abbreviated to M, D or C. For convenience, some example messages are shown in **Annex B**.*

***NOTE** For brevity, prefixes to protobuf enumerations have been omitted from tables, for example, the "status" message 'Info' enumerations, listed as 'INFO_NEW' and 'INFO_UNCHANGED' in status_report.proto, are abbreviated to 'NEW' and 'UNCHANGED' in **6.2.7.2**, **Table 62**.*

4.1 General

SAPIENT nodes shall support Transmission Control Protocol/Internet Protocol (TCP/IP) (RFC 1122[1)]) as a default transmission protocol. Additional support for other transport protocols is optional, but TCP/IP shall be supported as a minimum.

For reliable operation and fusion of information, all network devices shall synchronize to a central Network Time Protocol (NTP) server. All nodes shall be expected to synchronize with this at regular time intervals during system operation.

Google Protocol Buffers version 3[2)] shall be used for compliance with this BSI Flex.

Universal Unique Identifiers (UUIDs)[3)] and Universal Unique Lexicographically sortable Identifiers (ULIDs)[4)] shall be used for compliance with this BSI Flex. All SAPIENT nodes that are capable of sending SAPIENT messages shall have a UUID "node_id", and include it in the message wrapper.

***NOTE 1** Protobuf uses "UTF-8" encoding as standard.*

***NOTE 2** The accompanying .proto files use the namespace "sapient_msg" (or appropriate capitalization for each corresponding language, e.g., "SapientMsg" for C#, "uk.gov.sapientmsg" for Java). Java proto files would be named, e.g., "uk.gov.sapientmsg.alert.protos". This namespace name does not affect the binary encoding of the messages.*

1) See https://www.rfc-editor.org/rfc/rfc1122.
2) See https://developers.google.com/protocol-buffers.
3) See https://datatracker.ietf.org/doc/html/rfc4122.
4) See https://github.com/ulid/spec.

All messages shall conform to the message wrapper structure in **Table 1**.

Table 1 – Overall message wrapper structure

Type	Field name	Status	Protobuf field serial	Description
google.protobuf. Timestamp	timestamp	M	1	UTC time of the content of the message
string	node_id	M	2	The node ID of the sender of the message
string	destination_id	C	3	The node ID of the destination node
oneof	content	M	–	The actual SAPIENT message, as referenced below:
Registration	–	–	4	SAPIENT Registration message
RegistrationAck	–	–	5	SAPIENT Registration Acknowledgement message
StatusReport	–	–	6	SAPIENT Status Report message
DetectionReport	–	–	7	SAPIENT Detection Report message
Task	–	–	8	SAPIENT Task message
TaskAck	–	–	9	SAPIENT Task Acknowledgement message
Alert	–	–	10	SAPIENT Alert message
AlertAck	–	–	11	SAPIENT Alert Acknowledgement message
Error	–	–	12	SAPIENT Error message
string	additional_ information	D	13	Space for additional information

NOTE 3 *The "timestamp" field specifies the time of the content of the message rather than the transmission time of the message. For example, for detection messages this is the time the detection was made.*

NOTE 4 *The "conditional" field (destination_id) is conditional because the id of the destination node is not always known (e.g. on registration) but is provided once established,*

4.2 Message framing/structure

NOTE 1 *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 serialized over a continuous byte stream, each message shall have the length of the message in bytes added as a prefix to the message. This length shall take the form of a 32-bit (4-byte) little endian number (just the length of the actual message, not including the 4-byte prefix).

NOTE 2 *This differs from the standard Google specification.*

The protobuf messages shall be combined in a single "SapientMessage" message, with the following fields common to all messages: timestamp, source node ID, and discretionally, the destination node ID. This shall be followed by a single "One Of" message content field containing the remaining fields for each message type.

NOTE 3 *A value of zero is listed as 'UNSPECIFIED' in the enumeration tables in this document. If a mandatory enumeration field is not explicitly set to a valid value of 1 or greater the field has not been set correctly and this constitutes an error. This would appear to the receiver as a value of zero. A zero in a discretionary enumeration field is not an error because it simply indicates that the value has not been set by the edge node.*

4.3 Version-number string

The version-number string, embedded as a protobuf file option, shall be used to indicate the version of the standard that those .proto files support.

NOTE *The .proto file for the "SapientMessage" message contains a string representing the version of the standard that those .proto files support, which is used for information purposes (rather than as part of some formal version negotiation mechanism).*

4.4 Initialization

The message transfer sequence shall start with Initialization.

The Initialization process shall follow the sequence:

a) a node connects as a client to the message handling application and sends a registration message;

b) the message handling application sends a registration acknowledgement message to the node;

c) the node sends an initial status message to the message handling application;

NOTE *At this point the message handling application or fusion node may optionally store relevant data from the message.*

d) a sensor edge node begins activity detection using its default tasking (for those sensor edge nodes which have defaults); and

e) an effector node awaits further tasking.

4.5 Normal operation

If the fusion node defines a task for an edge node, that task shall override any node-generated default task previously defined.

NOTE *Example steps in the normal operation process are as follows. These refer to sensor edge nodes and are illustrative. The process may vary for different edge nodes with different levels of autonomy, responding to different real-world events.*

a) Sensor edge node performs activity detection.

b) Sensor edge node sends regular status messages to the message handling application.

c) Sensor edge node sends detection messages using current tasking to the message handling application.

d) Sensor edge node discretionally sends alert messages to the message handling application.

e) The message handling application or fusion node may store messages or the data within them.

f) Fusion node queries the database and reasons over/fuses the detections.

g) Fusion node sends detection messages containing fused track points to the message handling application.

h) Fusion node sends alert messages to the message handling application.

i) Fusion node sends task messages to edge nodes via the message handling application.

j) The message handling application or fusion node may store messages or the data within them.

k) UI queries the store to show the output from the fusion node.

4.6 Node alerts during normal operations

***NOTE 1** Node alert conditions are defined by node capabilities, not standardized. An example of an edge node-defined condition may be a sensor edge node deciding to do a calibration scan of an object or area.*

The alert process shall follow the sequence:

a) if the alert condition is true the alert message shall be sent by the node;

b) if the alert message requires a response from the fusion node, the fusion node shall respond by sending an alert response message to the node via the message handling application (if applicable);

c) if the fusion node chooses to deny permission to carry out the action, the fusion node shall respond with an alert response message with "status" field of value "reject"; and

d) if the fusion node grants permission to carry out the action, the fusion node shall respond with an alert response message with "status" field of value "accept".

***NOTE 2** If rejected, the "reason" field may be used to provide a strategy for retrying the action in future.*

4.7 Task messages

The task process shall follow the sequence:

a) the fusion node shall send the "task" message to the node via the message handling application (if applicable);

b) the edge node shall send a "task acknowledgement message to the fusion node via the message handling application (if applicable), confirming acceptance or otherwise of the task, if accepted, the node shall change behaviour as instructed by the task; and

c) if the task is a command-based task, on completion of the task the node shall return to its previous region-based task. If no default region-based task exists, after completion of the task the node shall stop performing a task until it receives a new task.

4.8 Node shutdown

In the event of a node shutdown the node shall send a status message with the system field set to "GOODBYE" to show impending loss of communication.

4.9 Lost connection

The fusion node and every edge node shall monitor the connection to the message handling application and shall attempt to reconnect if connection is lost.

If a node or fusion node is unable to connect to the message handling application, then it shall attempt to connect every 10 seconds (s) until it is successful.

If reconnection occurs within 2 minutes (min) of the node noticing loss of connection, re-sending a registration message shall not be necessary. If a longer time has passed, the ASM shall re-send a registration message.

***NOTE** Monitoring of the network connectivity is not covered by BSI Flex 335.*

4.10 Platforms

*COMMENTARY ON **4.10***

BSI Flex 335 v2.0 supports the grouping together of edge nodes on "platforms". The purpose of this grouping is to indicate to the fusion node a dependency between the edge nodes and the platform that the fusion node is to take into account when tasking the edge nodes or platform node. There are two use cases:

a) *where the platform can be pointed in different directions, for example a gimbal or turret, but maintains the same geographic location; and*

b) *where the platform can be moved to different geographic locations, for example, a crewed or uncrewed ground vehicle or air vehicle.*

In these use cases, the fusion node is not physically located on the platform; only edge nodes are physically based on the platform.

*To implement platform architecture, changes to BSI Flex 335 have been made in **Table 15** (to enable declaration of dependent nodes and reporting region), **Table 17** (node definition), **Table 18** (node type), **Table 25** (status report category) and **Table 26** (to add additional status reports concerning platforms).*

Where a fusion node is physically located on a platform the requirements of **4.11** shall be met.

***NOTE 1** The approach taken to implement "platforms" is to create a platform node that registers with a fusion node in the same manner that the edge nodes mounted on the platform do.*

The platform node registration message shall declare whether it is a 'POINTABLE_NODE' or a 'MOBILE_NODE' and shall declare the UUID node_ids of any nodes (edge node or platform nodes) that are dependent on the registering platform node.

***NOTE 2** This informs the fusion node of the nature (pointable or mobile) of the platform, and which (dependent) nodes will be affected by commands sent to the platform node.*

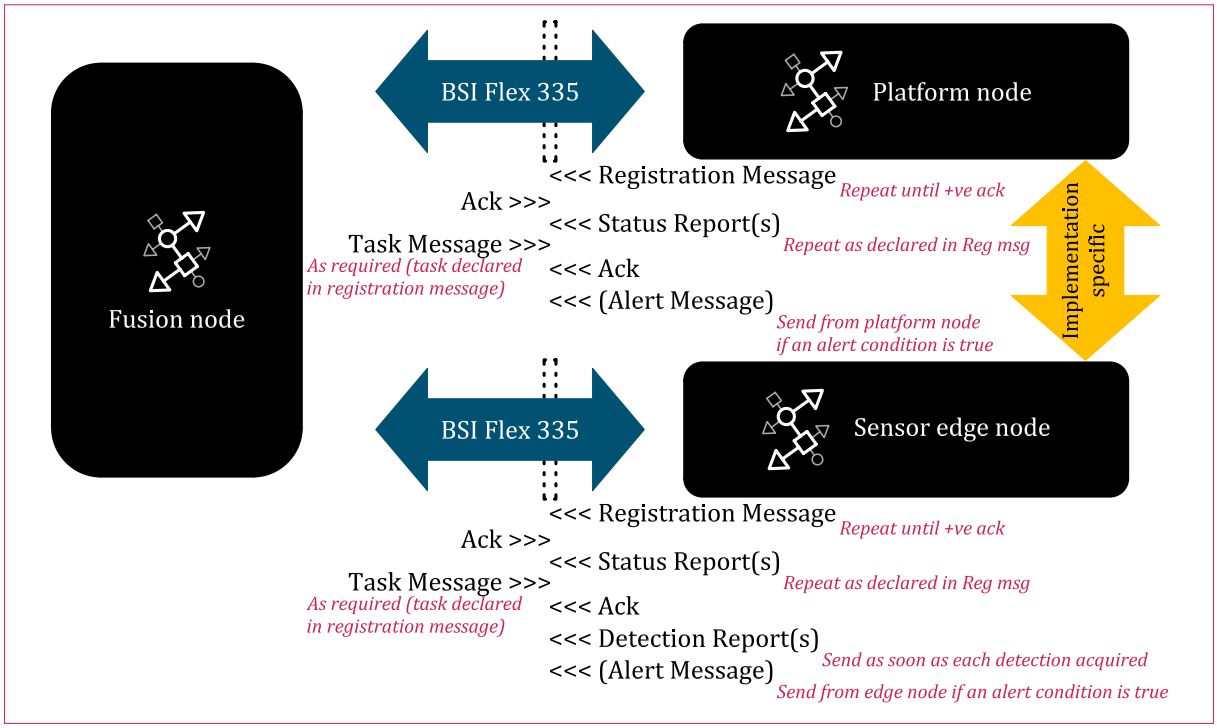
Platform nodes and dependent nodes shall declare the commands they are capable of accepting in their registration message.

***NOTE 3** Platform nodes are expected to declare commands relating to movement of the platform, such as "Look_At", "Move_To" (as appropriate). Dependent (sensor or effector) edge nodes are expected to declare commands relating to the intrinsic performance of the node, such as "Detection_Threshold", or "Mode_Change", for example. However, there might be implementation-specific reasons to vary from this advice.*

__NOTE 4 Figure 3__ shows the architecture of a SAPIENT platform and the messages sent between nodes. The architecture implies some implementation-specific interfaces between the dependent nodes and the platform node control systems. For example, dependent nodes may require information such as location, velocity, roll, pitch and yaw to apply corrections to their observations or to populate elements of their detection or status messages, if the nodes do not have their own organic capability to supply the necessary information. The platform node may require information from the dependent node(s) on the location of detections if the platform is capable of supporting the "Follow" command.

If platform type is not defined as mobile or pointable, it shall be assumed to be static.

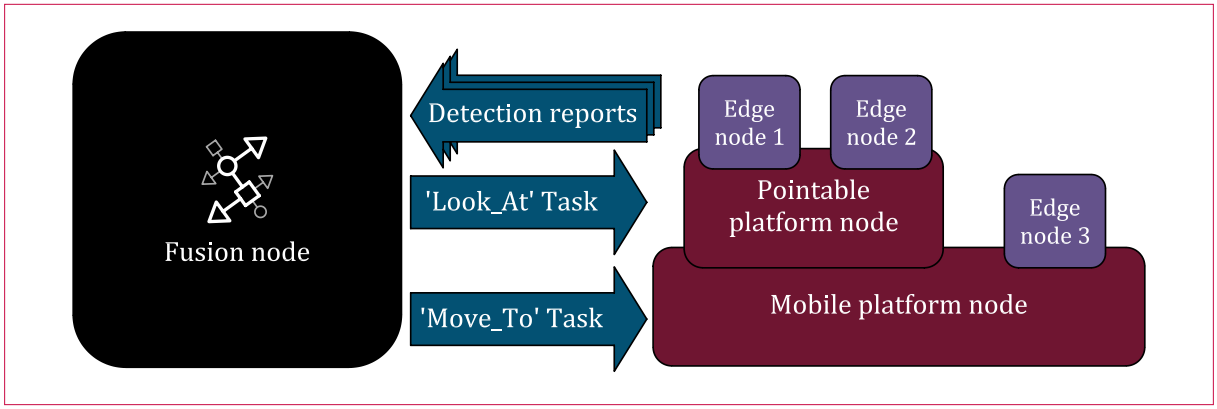
Figure 3 – Message sequence for a platform architecture



Complex dependencies shall be described using platforms stacked on platforms.

__NOTE Figure 4__ shows an example. In this case the mobile node declares the pointable node and edge node 3 in its registration message, and the pointable node declares edge nodes 1 and 2 in its registration message.

Figure 4 – An example of a complex platform node



4.11 Hierarchical architecture

*COMMENTARY ON **4.11***

A hierarchy of fusion nodes enables, for example:

a) fusion nodes that cover local regions to be aggregated by higher-level fusion nodes that cover larger areas; or

b) fusion nodes that control the edge nodes on a structure (e.g. a vehicle) to, themselves, be controlled by higher-level fusion nodes that control multiple structures.

In this architecture the lower-level fusion nodes behave (to higher-level fusion nodes) as if they are edge nodes. They register, send detection reports and are taskable as if they were edge nodes. This gives a 'level-agnostic' quality to nodes and allows the hierarchical architecture to be extensible over an arbitrary number of levels, according to the requirement of the application.

Fusion nodes that support being subordinate (child nodes) and are configured as such, shall publish and receive messages as edge nodes do. The capability of the fusion node, in terms of what functionality it can support, shall be declared in the registration message sent by the child fusion node to its parent, just as an edge node declares its capability in its registration message.

The following requirements shall be met for the minimum levels of functionality:

a) No hierarchical capability – no registration message shall be sent, no communication is required to a higher node.

b) Child fusion node supports detection – the child fusion node shall be capable of forming and sending a registration message to the parent. The registration message shall indicate the combined detection capabilities of the fusion node, as if the fusion node were an edge node with the aggregated detection capabilities of all of its actual edge nodes.

c) Child fusion node supports tasking – the child fusion node shall be capable of forming and sending a registration message to the parent. The registration message shall indicate the tasks and modes it supports, as if the fusion node were an edge node supporting the aggregated tasking capabilities of all its actual edge nodes.

***NOTE 1** To enable the required behaviour, a node type of fusion node has been added to **Table 18**, and changes have been made in **Table 60** (to enable a list of coverage areas for aggregation of fusion nodes that cover local regions).*

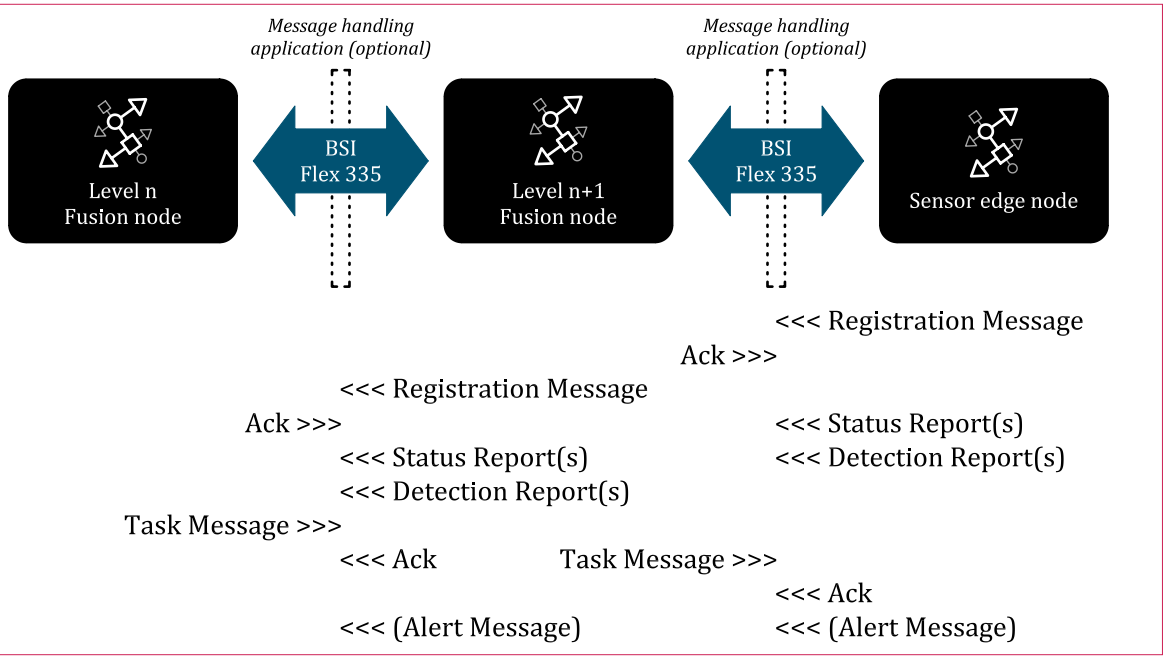
***NOTE 2** Hierarchical functionality can exist in static fusion nodes or in combination with platform nodes (see **4.10**).*

In the special case that a child fusion node physically resides on a platform (mobile or pointable) and within a hierarchy of fusion nodes, the child fusion node shall report a "NodeDefinition" of both "Fusion_Node" and platform type (i.e. "Mobile_Node" or "Pointable_Node") (see **Table 15**) in its registration message and declare what taskings it is capable of supporting.

***NOTE 3** The parent fusion node can then send appropriate commands to the child fusion node; the child fusion node autonomously determines how best to satisfy the request(s) of its parent and commands its platform node (on which it physically resides) if necessary.*

The message sequence for a hierarchical architecture shall follow **Figure 5**.

Figure 5 – Message sequence for a hierarchical architecture



This architecture shall be repeated for as many levels of fusion node as required in the system.

In a hierarchical architecture the highest level fusion node shall act as a 'server'. A mid-level fusion node shall open two connection channels, one that acts as a 'client' to the parent fusion node and one that acts as a 'server' to the lower-level nodes. The edge nodes shall act as a 'client'.

Edge nodes (and indeed lower level fusion nodes) may go on- and off-line, during operation; in this case the child fusion node shall update its coverage area using the status message (see **Table 60**).

NOTE 4 *As illustrated in* ***Figure 5****, the preferred sequence of registration is 'bottom-up'; i.e. the edge nodes register with the lowest level fusion node first. This is not mandatory, but avoids multiple updates of coverage area.*

5 Inner Message Types

*COMMENTARY ON CLAUSE **5***

A number of data structures are used in multiple messages. These have their own .proto files and are listed as follows:

a) location;

b) range bearing;

c) velocity;

d) associated detection; and

e) associated file.

5.1 General

The data structures listed in the commentary on Clause **5** shall be referred to as Inner messages.

***NOTE** Where the sections for inner and outer message types directly relate to a message or enumeration type within the .Proto files, the titles adhere to the following syntax, File message:File-submessage (e.g. Velocity message:ENUVelocity or Registration message:StatusReport-StatusReportCategory). This is in order to assist with location of a particular message type and associated fields within the .PROTO files. Sections that describe individual fields within message types do not follow this syntax for section headings.*

5.2 Location types

5.2.1 General

To support a variety of sensor types, location information shall be supported in a number of different real-world coordinate systems.

a) LAT/LONG and UTM coordinate systems shall be supported for global locations, also referred to as Cartesian locations.

b) Detections and status information shall also be supported in spherical coordinates (range/bearing/elevation) relative to a known node location.

Providing object locations in any of these formats shall be acceptable, depending on what the native format is of the individual edge node.

***NOTE 1** To keep this flexibility bounded, this should be limited to cartesian (x, y, z coordinates, together with a corresponding reference frame, either UTM or LAT/LONG) or spherical (range-bearing-elevation). Within each of these, a number of coordinate systems and datums/units are provided as enumerations.*

The coordinate system used for the different types of location shall conform to **Table 2**. Where supported by a node, the coordinate system used shall be declared in the registration message.

This coordinate system shall be used consistently for all messages that contain location data of that type. Non-selected coordinate systems shall be ignored entirely. For Cartesian coordinates, the node shall use the same reference frame (i.e. LAT/LONG or UTM) for all location types.

Table 2 – Coordinate system for different types of location information

Location type	Coordinate system
Detections	Any
Node location	Cartesian
Edge node field of view and coverage	Any – Range-bearing might be preferable for steerable edge nodes
Areas of obscuration	Any
Regions	Any – Cartesian preferred because regions may be independent of edge nodes

Any object sizes and speeds reported shall also be given in real-world units. Internal frames of reference shall be calibrated and translated into real-world values with associated errors if appropriate before being reported.

NOTE 2 *An example of an internal frame of reference is camera-centric, measured in terms of pixels.*

The fusion node shall task the individual nodes using the chosen coordinate system of the node as specified by the "TaskDefinition" section of the edge node registration message.

All azimuth bearings shall be reported relative to one of grid north, magnetic north or true north.

Grid north shall be used by default but magnetic north or true north can be specified in the edge node registration message.

Where messages require that either a location or range bearing be provided, a protobuf "one of" field is used to provide the options and the valid entries shall be as follows:

a) location is the location being reported; or

b) "RangeBearing" is the range bearing being reported.

NOTE 3 *Some messages require location list or range bearing cones.*

In these situations, a protobuf "OneOf" field, named "LocationOrRangeBearing" is used and the valid entries shall be as follows:

1) the message "LocationList"; or

2) the message "RangeBearingCone".

5.2.2 Location message:Location

5.2.2.1 General

The fields in the location message shall conform to **Table 3**.

The message "LocationList" shall be used to report a list of locations.

Table 3 – "Location" message information

Type	Field name	SAPIENT Field Status	Protobuf field serial	Description
double	x	M	1	Eastings up to 7 decimal places
double	y	M	2	Northings up to 7 decimal places
double	z	D	3	Altitude of object in metres up to 5 decimal places
double	x_error	D	4	Error of location up to 7 decimal places
double	y_error	D	5	Error of location up to 7 decimal places
double	z_error	D	6	Error of location up to 5 decimal places
string	utm_zone	D	9	Where a node is close to the edge of a UTM zone and needs to report a location in a different UTM zone from that declared in its registration message, this field may be used
LocationCoordinateSystem enum	coordinate_ system	M	7	Coordinate system used
LocationDatum enum	datum	M	8	Datum used

If "z" (altitude) is omitted from the message, this shall indicate altitude ambiguity, i.e. the object could be at any altitude – it is not possible to resolve the altitude.

5.2.2.2 Location message:Location-LocationCoordinateSystem

Details of the location coordinate system enumeration field shall conform to **Table 4**.

NOTE *Non-SI units have been deprecated.*

Table 4 – Valid values for the location coordinate system enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Coordinate system/units not defined
LAT_LNG_DEG_M	1	Latitude/Longitude in decimal degrees, altitude in metres
LAT_LNG_RAD_M	2	Latitude/Longitude in radians/metres
UTM_M	5	UTM with altitude in metres

Altitude shall be reported in metres above the location datum (see **Table 5**).

5.2.2.3 Location message:Location-LocationDatum

Values of the location datum enumeration field shall conform to **Table 5**.

Table 5 – Valid values for the location datum enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Datum not defined
WGS84E	1	WGS84 Ellipsoid
WGS84G	2	WGS84 Geoid

5.2.2.4 Location message:LocationList

Location elements that define the boundary polygon of a region or sensor field of view shall be defined by the values in **Table 6**.

Table 6 – "LocationList" message

Type	Field name	Status	Protobuf field serial	Description
Location	locations	M List	1	A list of locations used to define a region or field of view

5.2.2.5 Range_Bearing message:RangeBearing

A single point in space in spherical form shall be defined by the "RangeBearing" field. Messages including "RangeBearing" information shall conform to **Table 7**.

Table 7 – Information for the "RangeBearing" message field

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
double	elevation	D	1	Angle above/below horizontal in radians up to 4 decimal places
double	azimuth	D	2	Angle relative to north – based on node's concept of north – up to 4 decimal places
double	range	D	3	Distance from edge node to object in metres up to 2 decimal places
double	elevation_error	D	4	Error of location in radians up to 4 decimal places
double	azimuth_error	D	5	Error of location in radians up to 4 decimal places
double	range_error	D	6	Error of location in metres up to two decimal places
RangeBearing CoordinateSystem enum	coordinate_ system	M	7	Coordinate system used
RangeBearingDatum enum	datum	M	8	Datum used

If elevation is omitted from the message, this shall indicate elevation ambiguity; i.e. the object could be at any elevation – it is not possible to resolve the elevation.

5.2.3 Range_Bearing message:RangeBearing-RangeBearingCone

A cone or field of view in spherical form shall be defined by the "RangeBearingCone" field. Messages including information for the "RangeBearingCone" message shall conform to **Table 8**.

Table 8 – Message field information for the "RangeBearingCone" message

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
double	elevation	D	1	Angle above/below horizontal in radians up to 5 decimal places – if omitted then object assumed to be on the ground-plane
double	azimuth	D	2	Angle of detection relative to North – based on node's concept of north
double	range	D	3	Distance from edge node to object in metres up to 5 decimal places
double	horizontal_extent	D	4	Horizontal extent angle of region in radians up to 5 decimal places
double	vertical_extent	D	5	Vertical extent angle of region in radians up to 5 decimal places
double	horizontal_extent_error	D	6	Error in horizontal extent angle
double	vertical_extent_error	D	7	Error in vertical extent angle
double	elevation_error	D	8	Error of location in radians up to two decimal places
double	azimuth_error	D	9	Error of location in radians up to two decimal places
double	range_error	D	10	Error of location in metres up to 2 decimal places
RangeBearing CoordinateSystem enum	coordinate_system	M	11	Coordinate system used
RangeBearingDatum enum	datum	M	12	Datum used

5.2.4 Range_Bearing message:RangeBearing-RangeBearingCoordinateSystem

Messages including enumeration information for the "RangeBearingCoordinateSystem" field shall conform to the values in **Table 9**.

Table 9 – Valid values for the "RangeBearingCoordinateSystem enum" field

Type	Enumeration value	Description
UNSPECIFIED	0	Coordinate system/units not defined
DEGREES_M	1	Values in decimal degrees and meters
RADIANS_M	2	Values in radians and meters
DEGREES_KM	3	Values in decimal degrees and kilometres
RADIANS_KM	4	Values in radians and kilometres

5.2.5 Range_Bearing message:RangeBearing- RangeBearingDatum

Messages that contain enumeration information for the "RangeBearingDatum" field shall conform to **Table 10**.

Table 10 – Valid values for the "RangeBearingDatum enum" field

Type	Enumeration value	Description
UNSPECIFIED	0	Datum not defined
TRUE	1	True north
MAGNETIC	2	Magnetic north
GRID	3	Grid north
PLATFORM	4	'North' is the heading of the platform carrying the node

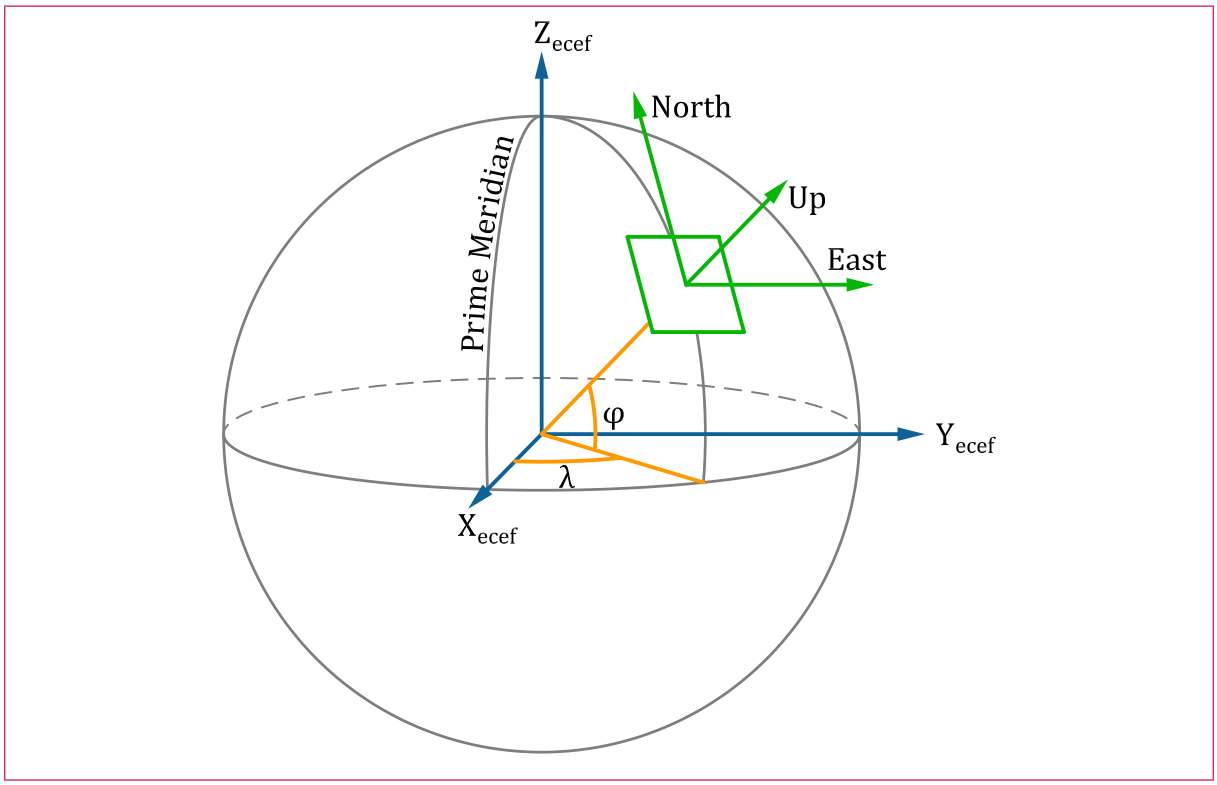
***NOTE** The datum for a bearing relative to a platform (**Table 10**) is currently not defined. This will be defined in the next edition of the standard.*

5.3 Velocity

5.3.1 General

Velocity shall be described using "ENUVelocity" – velocity as east rate, north rate and up rate: coordinates east rate, north rate, up rate, as shown in **Figure 6**.

Figure 6 – East, North, Up (ENU) frame of reference



***NOTE** While other frames of reference exist and space has been left for future additions, for this version of BSI Flex 335, only ENU should be used pending further discussions.*

5.3.2 Velocity message:ENUVelocity

***NOTE** This message provides velocity as a vector in global cartesian coordinates: [This is a revision of the velocity encoding used at the NATO NCIA Technical Interoperability Exercise 2022 (TIE22)[5)] as per the Location field but with Eastings, Northings and Up for clarity.] This aligns with the Location (Cartesian) coordinates system.*

[5)] See https://www.ncia.nato.int/about-us/newsroom/nci-agency-holds-natos-livetesting-counterdrone-exercise.html.

Messages that contain "ENUVelocity" field information shall conform to the field values in **Table 11.**

Table 11 – Information for the "ENUVelocity" field

Type	Field name	SAPIENT Field Status	Protobuf field serial	Description
double	east_rate	M	1	Velocity in the east-axis (x)
double	north_rate	M	2	Velocity in the north-axis (y)
double	up_rate	C	3	Velocity in the up-axis (z)
double	east_rate_error	C	4	Error in the east-axis (x) value
double	north_rate_error	C	5	Error in the north-axis (y) value
double	up_rate_error	C	6	Error in the up-axis (z) value

Velocity units shall be expressed in SI.

Conditional fields shall be provided if available.

5.4 Associated_File message:AssociatedFile

Messages that contain information for the "AssociatedFile" field shall conform to the values in **Table 12**.

Table 12 – "Associated file" message field information

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
string	type	M	1	Type of file e.g. image
string	url	M	2	URL to the media. Typically the sensor edge node identifier shall be included as a sub folder in the URL

***NOTE** The type of file field is retained as a string in this version of BSI Flex 335 and should define a file type in MIME format, for example "image/jpeg" or "video/mpeg4". In future versions of BSI Flex 335 this will be amended to be an enumeration type defining a set of file types in MIME format.*

5.5 Associated_detection message:AssociatedDetection message

*COMMENTARY ON **5.5***

This subclause is a list of the IDs of detection reports with different object IDs related to the Detection Report in question. For example, for a sensor edge node 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 are 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 might be particularly useful where a sensor edge node can provide many different but related types of information about an object, e.g. when a person is carrying multiple emitting devices.

5.5.1 General

Messages that contain "AssociatedDetection" information shall conform to the field values in **Table 13**.

Table 13 – "Associated detection" message field information

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
google.protobuf. Timestamp	timestamp	C	1	Timestamp of associated detection
string	node_id	M	2	Unique identifier of fusion node that was the source of the detection
string	Object_Id	M	3	Unique identifier of the detection
AssociationRelation enum	association_type	D	4	Relationship of associated detection to this detection

If used by a fusion node, the "associated_detection" message shall represent a list of individual sensor edge node detections that were used to generate a fused detection.

***NOTE** Tracks of objects should not be linked like this. Tracks should be a series of detection reports with the same object ID.*

5.5.2 Associated_detection message:AssociatedRelation

Messages that contain information for "AssociatedRelation" enumeration shall conform to the values in **Table 14**.

Table 14 – Valid values for the "Associated relation" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	No relation – used to support default values in protobuf
NO_RELATION	1	The associated detection is not related to this detection
PARENT	2	The associated detection is a parent of this detection
CHILD	3	The associated detection is a child of this detection
SIBLING	4	The associated detection is a sibling of this detection

6 Outer message types

6.1 General

All of the outer messages types (registration, status, detection, alert, error and corresponding Acknowledgements) shall be contained in an overarching wrapper defined in the "sapient_message.proto" file.

6.2 Registration

6.2.1 General

As part of the system initialization, each node shall produce a registration message defining its capabilities. This shall include:

a) node type;

b) interface version the node implements;

c) node name;

d) list of node capabilities used to give extra detail on what modalities and functions the node contains;

e) status message definition; and

f) a list of definitions of supported modes.

The status message definition shall include:

1) a description of the contents of its status message;

2) how node locations are reported in the status message; and

3) how frequently the status message is to be transmitted.

The mode definition shall include:

i) a description of the mode and node behaviour when in that mode;

ii) mode timing information;

iii) definition of detection messages in use with the mode; and

NOTE 1 *Where a node is a sensor, it should list what it is able to detect, track and/or classify.*

iv) a list of how the node is able to interact with the fusion node, e.g. tasks it is able to accept whilst in this mode.

The message shall include the performance implications of all detection modes. Performance shall be specified in terms of either geometric (location) error, detection performance or classification performance. General components of the registration shall conform to **Table 15**.

NOTE 2 *The node type enumeration allows the modality or type of node that is registering to be specified.*

Table 15 – General structure of a registration message

Type	Field name	Status	Protobuf Field Serial	Description
NodeDefinition	node_definition	M List	1	Node type defines the broad category of node registering to allow the fusion node to handle the node appropriately
string	ICD_version	M	2	ICD version implemented by the node
string	name	D	3	Discretionary name for individual instance of node, e.g. car park camera
string	short_name	D	4	two-digit abbreviation of name, e.g. C1 for camera 1
Capability	capabilities	M List	5	List of node capabilities, used to provide guidance to fusion nodes on what modalities/functionality a node contains
StatusDefinition	status_definition	M	6	—
ModeDefinition	mode_definition	M List	7	—
string	dependent_nodes	D List	8	Where the tasking of one node may also affect other nodes, e.g. a mobile platform or gimbal, the UUID of the dependent nodes is reported here
ConfigurationData	config_data	M List	9	Configuration data lists information to manage nodes and their configuration status

The field “ICD_version” shall use the following form:

```
"BSI Flex 335 vX.Y"
```

where X is the major version and Y is the minor version (e.g. this version is “BSI Flex 335 v2.0”).

6.2.2 Registration acknowledgement message protocol

The registration message shall be sent by the node to the message handling application before any other messages. No other messages shall be sent until a registration acknowledgement message or error message is received in response. The format of the registration acknowledgement message shall conform to **Table 16**.

Table 16 – "Registration acknowledgement" message format

Type	Field name	Status	Protobuf Field Serial	Description
boolean	acceptance	M	1	This field takes a value of "FALSE" if the registration is rejected and a value of "TRUE" if the registration is accepted
string	ack_response_reason	D List	3	This field can describe the (operational) reason(s) for rejecting the registration

The "destination_id" field in **Table 1** shall be populated in a registration acknowledgement message.

The registration message shall be sent when the node connects or reconnects to the system, or when requested by a fusion node.

NOTE *It is valid for a fusion node to request re-sending of a registration message at any time.*

If the node does not receive a response within 30 s of sending the registration message, the node shall close the existing connection, open a new connection to the message handling application and send the message again.

The registration acknowledgement message shall be used by the message handling application to acknowledge to the node that registration has or has not occurred. The registration acknowledgement message shall be used where the registration has been rejected for an operational reason, for example, insufficient resource to support the node. If a registration message contains an error, a SAPIENT "error" message shall be sent, and the registration acknowledgement message shall not be sent.

6.2.3 Location registration information

The registration message shall be used to declare the real-world coordinate system that the node is going to use to provide location information to the message handling application in all subsequent messages. Once a node has declared the type of coordinate system it is using, it shall consistently use this system. The fusion node shall consistently respond to the node in the same system.

NOTE *This places an obligation on fusion node suppliers to support all coordinate systems defined in this document.*

6.2.4 Node identification

Each node shall use a single, unique node identifier in all messages. This identifier shall be a UUID v4.

The "node_id" value shall define the identity of the source of the message. The node shall use the "node_id" value sent in the initial registration message for all subsequent messages.

The "destination_id" field shall define the identity of the destination of the message.

6.2.5 Detailed description

*COMMENTARY ON **6.2.5***

This subclause contains specific implementation detail that is to be adopted in a registration message.

6.2.5.1 Registration message:NodeDefinition

The "NodeDefinition" message shall conform to the values in **Table 17**.

***NOTE** The "NodeDefinition" message is a list of generic types of sensing node or effector node types that the node supports.*

Table 17 – "NodeDefinition" message

Type	Field name	Status	Protobuf field serial	Description
NodeType enum	node_type	M	1	The generic type(s) of sensor or effector supported by the Node
string	node_sub_type	D	2	Additional information refining the node_type field

6.2.5.2 Registration message:NodeDefinition-NodeType

Node types shall be specified if they are relevant to the node that is registering, and the values for these shall conform to **Table 18**.

***NOTE 1** The node type enumeration allows the modality or type of node that is registering to be specified.*

Table 18 – Node type enumeration

TYPE	Enumeration value	Description
UNSPECIFIED	0	No node type set
OTHER	1	Used to represent a node type not included in this enumeration
RADAR	2	Used to represent a node that uses/is a radar
LIDAR	3	Used to represent a node that uses/is a lidar
CAMERA	4	Used to represent a node which contains/is a camera
SEISMIC	5	Used to represent a node that uses ground vibrations
ACOUSTIC	6	A system based on acoustic signals, e.g. microphone
PROXIMITY_SENSOR	7	A system that can detect the local presence of objects, e.g. trip-wire
PASSIVE_RF	8	A system based on the passive reception of Electromagnetic radiation
HUMAN	9	A human acting as part of the SAPIENT system
CHEMICAL	10	A system that can detect chemical signatures

Table 18 – Node type enumeration *(continued)*

TYPE	Enumeration value	Description
BIOLOGICAL	11	A system that can detect biological signatures
RADIATION	12	A system that can detect radioactive signatures, e.g. Geiger counter
KINETIC	13	A system that has a physical effector, e.g. a missile
JAMMER	14	A radio frequency transmitter that disrupts other systems using power
CYBER	15	A system that uses network/communications protocols to detect or effect other systems
LDEW	16	Laser Directed Energy Weapon
RFDEW	17	Radio Frequency Directed Energy Weapon
MOBILE_NODE	18	A (platform) node that can be moved to a new location
POINTABLE_NODE	19	A node that can be pointed in different directions, e.g. a gimbal
FUSION_NODE	20	A (child) Fusion Node that has registered with a parent Fusion Node

NOTE 2 *The term "POINTABLE_NODE" refers to the type of Node; this should not be confused with the term 'STEERABLE' in* ***Table 29****, which refers to the mode scan type.*

6.2.5.3 Registration message:Capability

The capability field shall define the capabilities that a node can provide such as type of electronic signal detection. If these fields are to be reported on an ongoing basis they shall be reported as part of the status list in the status report. The capability field shall conform to **Table 19**.

Table 19 – "Capability" field types

Type	Field name	Status	Protobuf Field Serial	Description
string	category	M	1	The category of field to report, e.g. ESM, Jammer
string	type	M	2	Description of the capability, e.g. maximum transmit power
string	value	D	3	Value of this parameter, e.g. 50
string	units	D	4	A description of the units or valid values that are to be reported, e.g. dB. Units shall be expressed in SI.

6.2.5.4 Registration message:StatusDefinition

The status definition section of the registration message shall define which fields are to be present in status report messages generated by the node and what format and units the fields are to use. Status definition fields shall conform to **Table 20**.

Table 20 – "Status definition" fields

Type	Field name	Status	Protobuf Field Serial	Description
Duration	status_interval	M	1	Time interval between regular Status Report messages being issued by the node
LocationType	location_definition	D	2	Define the Coordinate system to use for node location, e.g. UTM or LAT/LONG. "RangeBearing" is not a valid value for node location.
				If providing edge node field of view, define the coordinate system and units to use
LocationType	coverage_definition	D	3	If providing edge node coverage, define the coordinate system and units to use
LocationType	obscuration_definition	D	4	If providing edge node obscuration regions, define the coordinate system and units to use
StatusReport	status_report	D List	5	–
LocationType	field_of_view_definition	D	6	If providing sensor field of view, define the Coordinate system and units to use

6.2.5.5 Registration message:Duration

The duration field shall specify the time interval between regular status report messages being issued by the edge node. The duration field shall conform to the values in **Table 21**.

NOTE 1 *If a status report is not received within three times this specified interval, the fusion node can assume this edge node has been disabled, turned off or lost communication.*

Table 21 – "Duration" fields

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
TimeUnits enum	units	M	1	A description of the units, e.g. HOURS, MILLISECONDS, SECONDS (see **Table 22**);. typically SECONDS
float	value	M	2	Typically between 1 and 60

Values for "TimeUnits" shall conform to the values in **Table 22**.

Table 22 – Units of time

Type	Enumeration value	Description
UNSPECIFIED	0	Time units not specified
NANOSECONDS	1	Used to express a duration in nanoseconds
MICROSECONDS	2	Used to express a duration in microseconds
MILLISECONDS	3	Used to express a duration in milliseconds
SECONDS	4	Used to express a duration in seconds
MINUTES	5	Used to express a duration in minutes
HOURS	6	Used to express a duration in hours
DAYS	7	Used to express a duration in days

NOTE 2 *Non-SI units have been deprecated.*

6.2.5.6 Registration message:LocationType

NOTE *Location type defines the format and units for each field that can provide location information.*

Whilst it is valid for a sensor edge node to provide some fields in spherical ("RangeBearing") coordinates and others in cartesian (UTM, LAT/LONG) coordinates, UTM and LAT/LONG coordinates systems shall not both be used within the same sensor edge node.

"LocationDatum" shall be used to specify coordinate system datum in use, typically WGS84[6)].

"RangeBearingDatum" shall be used to specify the north datum in use.

Location type information in a registration message shall conform to the values in **Table 23**.

Table 23 – "Location" type fields

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
oneof	coordinates_oneof	M	1, 2	Either "Location" or "RangeBearing" coordinate system and units being reported
oneof	datum_oneof	M	3, 4	Either the "Location" or "RangeBearing Datum" being reported
string	zone	C	5	If using UTM this is the UTM Zone, e.g. 30U; otherwise omit

6) See https://gisgeography.com/wgs84-world-geodetic-system/.

6.2.5.7 Registration message:StatusReport

The status report shall list the fields to be included in the registration message. Only fields relevant to the particular edge node shall be included; all others shall be omitted.

NOTE 1 *These fields in the registration message are distinct from status messages.*

NOTE 2 *For example, a radar-based sensor edge node would not report on its exposure or white balance, and a mains-powered edge node would not report battery level.*

Status report information in a registration message shall conform to **Table 24**.

Table 24 – Status report fields

Type	Field name	SAPIENT Field Status	Protobuf field serial	Description
StatusReportCategory enum	category	M	1	The category of field to report. This maps to the other fields of the status report message i.e. sensor, power, mode, status
string	type	M	2	The type or name of the information being provided
string	units	D	3	A description of the units or valid values that will be reported
bool	on_change	D	4	If TRUE, only report when the value changes. If FALSE, always report it

6.2.5.8 Registration message:StatusReport-StatusReportCategory

The status report category enumeration shall allow the category of a status report to be specified and the values for these shall conform to **Table 25**.

Table 25 – "Status report" category

Type	Value	Description
UNSPECIFIED	0	Status report category not defined
SENSOR	1	Means a report about the sensor edge node's performance
POWER	2	Means a report about the node's current power situation
MODE	3	Means a report about the node's current mode
STATUS	4	Means status fields can be reported

6.2.5.9 Status report values

Status values in a status report message (see **Table 60**) shall conform to **Table 26**. Type and unit fields shall be consistent with those provided in the main status messages.

Table 26 – Status report – status values

Category	Type	Units
sensor	not_detecting	Node_Moving, Node_Stopped, Node_Paused (Note: Node_Stopped or Node_Paused are reported when a node has received a 'Stop' or 'Pause' command, rather than the node being stationary)
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	Stationary, Slewing, Following, Stuck, Moving, Stopped (*Note: Moving and Stopped are deprecated in Flex 335 v2.0*)
status	PD	Probability
status	FAR	Probability
status	Platform	Stationary, Transiting, Slewing, Following, Stuck, Drifting

NOTE 1 *The "units" column in* ***Table 26*** *is retained as a string in this version of BSI Flex 335, although it defines a number of enumeratable types. In future versions of BSI Flex 335 this will be amended to be a specific set of enumeration types.*

NOTE 2 *The platform "status field" meanings are:*

Stationary – The mobile or pointable platform is not moving
Transiting – The mobile platform is moving to a new location in response to a tasking
Slewing – The pointable platform is moving to point in a different direction in response to a tasking
Following – The platform is moving to follow a designated object
Stuck – Mechanical or environmental conditions prevent movement despite power being applied
Drifting – Despite desiring to be stationary, the platform is moving with no, or limited, control.

6.2.5.10 Registration message:ModeDefinition

The mode definition fields of a "registration" message shall conform to the values in **Table 27**.

Table 27 – "Mode definition" fields

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
string	mode_name	M	1	Name of mode, e.g. default, follow
ModeType enum	mode_type	M	2	Type of mode (permanent, temporary)
string	mode_description	D	3	Free text description
Duration	settle_time	M	4	Time to settle to normal performance
Duration	maximum_latency	D	5	Time to wait before re-attempting communication
ScanType enum	scan_type	D	6	Unset, fixed, scanning, steerable. If omitted, assume fixed.
TrackingType enum	tracking_type	D	7	None, tracklet, track, "TrackWithReID". If omitted, assume none
Duration	duration	D	8	Duration of temporary mode before reverting to original mode
ModeParameter	mode_parameter	D List	9	A list of descriptive parameters for the mode that are not reported otherwise.
DetectionDefinition	detection_definition	C List	10	Conditional on whether the Edge Node creates detections, i.e. is not an effector
TaskDefinition	task	M	11	Defines region and parameters of a task, as well as how many tasks can run concurrently

Only one mode shall be default.

6.2.5.11 Registration message:ModeDefinition-ModeType

The mode type field shall define the mode type and conform to **Table 28**. If omitted, it shall assume the value of permanent.

NOTE *Permanent means that the edge node is set to stay in this mode until tasked otherwise. Temporary means that the edge node reverts to its previous mode after this mode is finished.*

Table 28 – Valid values for the "ModeType" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Mode type not specified
PERMANENT	1	Mode is continuous and does not change until the node is re-tasked
TEMPORARY	2	Mode is temporary and reverts to original mode once the mode's duration is reached
DEFAULT	3	The default mode is permanent until changed

6.2.5.12 Registration message:ModeDefinition-ScanType

The "ScanType" field shall define the edge node field of view behaviour. If omitted from the registration the value of "Fixed" shall be assumed. "ScanType" values shall conform to **Table 29**.

NOTE *Fixed means a field of view that does not change and all areas are observed at all times, e.g. a fixed video sensor. Scanning means that the sensor edge node only observes over part of its field of view at any time, e.g. a rotating radar. Steerable means that the edge node can be steered to point at a location, e.g. a Pan-Tilt-Zoom Camera.*

Table 29 – Valid values for "ScanType" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Scan type not specified
FIXED	1	Field of view does not change and all areas are observed at all times
SCANNING	2	Mode has a moving field of view that only detects part of a region at any given time
STEERABLE	3	Mode has a field of view that can be steered to point at a location

6.2.5.13 Registration message:ModeDefinition-TrackingType

The "TrackingType" field shall define the sensor edge node tracking capabilities and shall conform to the values of **Table 30**. If omitted, it shall be assumed that there are none.

NOTE *None means there is no data association between detections of the same object. A unique "object_id" is generated by the sensor edge node for each detection. TRACKLET means the sensor edge node attempts to maintain the "object_id" between detections of the same object but no attempt is made to join broken tracks. TRACK means that the sensor edge node applies a tracking algorithm to allow consistent "object_id" even behind occlusions. TRACK_WITH_RE_ID means that the sensor edge node can provide tracks that can be re-associated based on features.*

Table 30 – Valid values for "TrackingType" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	No data association between detections of the same object. A unique object_id is generated for each detection
NONE	1	The sensor edge node is not capable of making associations between detections of the same object
TRACKLET	2	The sensor edge node attempts to persist an object_id between detections of the same object, but does not attempt to join broken tracks
TRACK	3	The sensor edge node is able to persist an object_id between detections of the same object, even when tracks are broken
TRACK_WITH_RE_ID	4	The sensor edge node is able to provide tracks that can be re-associated based on features

6.2.5.14 Registration message:ModeDefinition-ModeParameter

The mode parameter field shall comprise of a list of descriptive parameters for the mode that are not reported otherwise.

Nodes requiring permission [**4.6**c) and d)] shall include in the registration message:

a) modeParameter type = "PermissionToChange"; and

b) value = "Required".

NOTE *These might be useful for the fusion node to determine the level of autonomy within the edge node. This is particularly relevant to scanning or steerable edge nodes that may alter their behaviour such as following a target or viewing multiple zones.*

Mode parameter fields in a registration message shall conform to **Table 31**.

Table 31 – "Mode parameter" field

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
String	type	M	1	e.g. Self-Adaptation
String	value	M	2	e.g. ROI, Range

6.2.5.15 Registration message: ConfigurationData

*COMMENTARY ON **6.2.5.15***

The configuration data sub-message allows a node to report useful information for managing the configuration of nodes, such as the manufacturer/model of a node, serial number, and hardware and software version numbers. Where a node is designed to consist of sub-components that can be added, removed or changed to modify the overall performance or capability of the node, it is possible to use this message to create a tree-like structure of sub-component configuration data.

The first (top-level) element of the list shall be the component that interfaces the node to the external SAPIENT system. Configuration data fields in a registration message shall conform to the values in **Table 32**.

Table 32 – Registration – "ConfigurationData" field

Type	Field name	SAPIENT Field Status	Protobuf field serial	Description
string	manufacturer	M	1	The name of the manufacturer of the node
string	model	M	2	The name of the product of the node
string	serial_number	C	3	The serial number of the node. Populated for nodes containing physical hardware. Pure software nodes may interpret this field as a product or license number
string	hardware_version	C	4	Populated for nodes containing physical hardware
string	software_version	C	5	Populated for nodes running software
ConfigurationData	sub_components	D List	6	Configuration data for sub-components can be specified here

6.2.5.16 Registration message:DetectionDefinition

Only fields relevant to the particular sensor edge node shall be included in the detection message while in the detection definition mode. Detection definition fields in a registration message shall conform to the values in **Table 33**.

Table 33 – "Detection definition" field

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
LocationType	Location_type	M	1	Location units to be used by detection messages in **Table 14**
DetectionPerformance	detection_performance	D List	2	List of performance measures useful to the fusion node
DetectionReport	detection_report	D List	3	Detection ID, location, primary information of detection report
DetectionClassDefinition	detection_class_definition	D List	4	Classes able to be reported in this detection report
BehaviourDefinition	behaviour_definition	D List	5	Behaviours able to be reported in the detection report
VelocityType	velocity_type	D	6	Defines the format and units for each field that can provide velocity information
GeometricError	geometric_error	D	7	List of location error characterizations to allow the fusion node to understand the detection performance of the sensor edge node

6.2.5.17 Registration message:DetectionDefinition-GeometricError

This field shall provide a list of location error characterizations to allow the fusion node to understand the detection performance of the sensor edge node. Geometric error detection definition fields in a registration message shall conform to values in **Table 34**.

Table 34 – "Detection definition geometric error" field

Type	Field name	Status	Protobuf Field Serial	Description
String	type	M	1	For example, Standard deviation
String	units	M	2	Units expressed in SI e.g. metres
String	variation_type	M	3	For example, Linear with range, squared with range
PerformanceValue	performance_value	D List	4	A list of types and values

6.2.5.18 Registration message:DetectionDefinition-DetectionReport

This field shall list the elements to be included in the detection message. Only fields relevant to the particular sensor edge node shall be included. All others shall be omitted. Detection report fields in a registration message shall conform to **Table 35**.

Table 35 – "Detection report" field

Type	Field name	Status	Protobuf Field Serial	Description
Detection ReportCategory enum	category	M	1	The category of field to report.
String	type	M	2	The type or name of the information being provided
String	units	M	3	A description of the units or valid values that will be reported. Units expressed in SI
Bool	on_change	D	4	If True, only report when the value changes. If omitted or false, always report it

6.2.5.19 Registration message:DetectionDefinition-DetectionReportCategory

The detection report category field shall conform to the values in **Table 36**.

Table 36 – Valid values for "Detection report category" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Detection report category not specified
DETECTION	1	Detection fields
TRACK	2	Track Info fields
OBJECT	3	Object Info fields
SIGNAL	4	Signal Fields

6.2.5.20 Registration message:DetectionDefinition – DetectionReport

The 'track_info' and 'object_info' fields of the detection message (see **Table 69**) shall conform to **Table 37**. The type and units fields shall be consistent with those provided in the detection report.

Table 37 – Detection report – "Track object info" value

Category	Type	Units	"DetectionReport" field
track	Confidence	probability	trackObjectInfo, "confidence"
object	dopplerSpeed	m/s	objectInfo, "dopplerSpeed"
object	dopplerAz	radians	objectInfo, "dopplerAz"
object	majorLength	metres	objectInfo, "majorLength"
object	majorAxisAz	radians	objectInfo, "majorAxisAz"
object	minorLength	metres	objectInfo, "minorLength"
object	height	metres	objectInfo, "height"

6.2.5.21 Registration message:DetectionDefinition-DetectionClassDefinition

Registration messages that contain detection class definition fields shall conform to values in **Table 38**.

Table 38 – "Detection class definition" field

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
ConfidenceDefinition enum	confidence_definition	D	1	Single Class, Multiple Class. Omit if not providing confidence
PerformanceValue	Class_performance	D List	2	—
ClassDefinition	Class_definition	D List	3	—
TaxonomyDockDefinition	Taxonomy_dock_ definition	D List	4	List of docking-points for taxonomy extensions

6.2.5.22 Registration message:DetectionDefinition-ConfidenceDefinition

The class performance field shall be a discretionary list of classification performance measures conforming to the values in **Table 39**.

Table 39 – Valid values for "confidence definition enumeration" field

Type	Enumeration value	Description
UNSPECIFIED	0	Confidence definition not specified
SINGLE_CLASS	1	Sensor edge node only provides confidence for the most likely class
MULTI_CLASS	2	Sensor edge node provides confidence for all classes

6.2.5.23 Registration message:DetectionDefinition-detection_performance

The detection performance field shall provide a list of performance measures useful to the fusion node. It shall only be provided if available. Detection performance in a registration message shall conform to values in **Table 40**.

Table 40 – "Definition detection performance" value field

Type	Field name	SAPIENT Field Status	Protobuf Field Serial	Description
string	type	M	1	Type of performance measure being defined, e.g. rotation speed
string	units	M	2	Units of performance measure being defined, e.g. degrees per second
string	unit_value	M	3	Typical value of the units of performance measure being defined, e.g. 2.5
string	variation_type	D	4	Variation description of performance measure being defined, e.g. linear

6.2.5.24 Registration message:DetectionDefinition – ClassDefinition

This field shall define the structure in which object classes and subclasses are provided by this sensor edge node. Class definition in a registration message shall conform to values in **Table 41.**

NOTE *Subclasses allow a hierarchy to define an object more narrowly.*

All classes shall be mutually exclusive and conform to the taxonomy in Clause **7**.

Table 41 – "Detection definition – class definition" field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Type of class being defined
string	units	D	2	Units of the confidence of the class being reported
Sub-class	sub_class	D List	3	Definitions of any subclasses

6.2.5.25 Registration message:DetectionDefinition -SubClass

The sub-class definition of the detection definition field of a registration message shall conform to the values in **Table 42**.

Table 42 – "Sub-class" definition of the detection definition field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Type of class being defined
string	units	D	2	Units of the confidence of the class being reported
int32	level	M	3	The level of the subclass (top level class would be 0, the first subclass would be 1, etc.)
Sub-class	sub_class	D List	4	Definitions of any subclasses

6.2.5.26 Registration message:DetectionDefinition-DetectionClassDefinition-TaxonomyDockDefinition

*COMMENTARY ON **6.2.5.26***

*This field defines the docking-points for dynamically defined taxonomy extensions (see Clause **7**).*

Namespaces are used as unique identifiers of both the new class and the existing class that the new taxonomy is a member of.

Docking points shall be a previously defined class, either a class from the SAPIENT core taxonomy, or a previously defined extension.

***NOTE 1** In the case of docking into the SAPIENT core taxonomy, the namespace would be "sapient_core".*

The fields for a taxonomy dock definition shall conform to **Table 43**.

Table 43 – "Taxonomy dock" definition field for taxonomy extensions

Type	Field name	Status	Protobuf Field Serial	Description
string	Dock_class_namespace	D	1	The namespace of the class that the new taxonomy is a member of. If omitted, assume the namespace is 'sapient_core'. However, do not omit if omission would create ambiguity.
string	Dock_class	M	2	The class that this new taxonomy is a member of; this shall be a previously defined class, either a class from the SAPIENT core taxonomy, or a previously defined extension.
Extension sub-class	Extension_sub_class	D List	3	List of subclasses of the Dock_class.

***NOTE 2** Example messages are provided in **Annex B**.*

6.2.5.27 Registration message:DetectionDefinition-DetectionClassDefinition-TaxonomyDockDefinition-ExtensionSubclass

*COMMENTARY ON **6.2.5.27***

*This field defines a new subclass for dynamically defined taxonomy extensions (see Clause **7**).*

A namespace shall be used as a unique identifier for the new taxonomy, i.e the combination of the namespace and the class name shall be unique.

***NOTE 1** The purpose of the namespace is to remove ambiguity, thus ensuring unique class names, eg. the class name 'Tanker' might be ambiguous, but the combination of https://discover.dtic.mil/thesaurus/ (the namespace) and Tanker (the class name in that namespace), is unique.*

Taxonomy extensions shall dock into existing classes in the SAPIENT core taxonomy. The level of the new class shall be implied from its docking point (i.e. the level is the docking point of the new class + 1).

The fields for an extension subclass definition shall conform to **Table 44**.

Table 44 – "Subclass definition for taxonomy extension" field

Type	Field name	Status	Protobuf Field Serial	Description
string	Subclass_namespace	M	1	Namespace of taxonomy. If taxonomy resource is not an internet resource, this can be a locally agreed namespace.
string	Subclass_name	M	2	Name of the new class. If taxonomy resource is an internet resource, this can be a URI
string	units	D	3	Units of the confidence of the class being reported

***NOTE 2** Example messages are provided in **Annex B**.*

6.2.5.28 Registration message:DetectionDefinition – Behaviour

This field shall define the structure as to how object behaviour or activity is provided for a sensor edge node. This field shall conform to the values in **Table 45**. All behaviours shall conform to the taxonomy in Clause **7**.

Table 45 – "Behaviour" definition field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Type of behaviour being defined
string	units	D	2	Units of the confidence of the behaviour being reported

6.2.5.29 Registration message:VelocityType

*COMMENTARY ON **6.2.5.25***

This subclause defines the format and units for each field that can provide velocity information.

Whilst it is valid for a sensor edge node to provide some fields in spherical ("RangeBearing") coordinates and others in cartesian (UTM, LAT/LONG) coordinates, UTM and LAT/LONG coordinate systems shall not be used within the same sensor edge node.

Velocity type information in a registration message shall conform to **Table 46**.

Table 46 – "Velocity type" field

Type	Field name	Status	Protobuf Field Serial	Description
oneof	velocity_units_oneof	M	4	Specifies that velocity is provided only in ENU (East, North, Up) units
oneof	datum_oneof	M	6,7	Either the Location or RangeBearing Datum of the velocity being reported
string	zone	C	8	If using UTM this is the UTM Zone, e.g. 30U; otherwise omit

6.2.5.30 Registration message:VelocityType-ENUVelocityUnits

The velocity units that define the ENU (East, North, Up) frame of reference described in **5.3.1** shall conform to the values in **Table 47**.

Table 47 – "ENU Velocity units" field

Type	Field name	Status	Protobuf Field Serial	Description
SpeedUnits enum	east_north_rate_units	M	1	Units that velocity is reported for in the east axis and north axis
SpeedUnits enum	up_rate_units	D	2	Units that velocity is reported for in the up axis

6.2.5.31 Registration message:DetectionDefinition-SpeedUnits

The Speed Units referenced in **6.2.5.30** shall conform to the values in **Table 48**.

Table 48 – "Speed units" field

Type	Enumeration value	Description
UNSPECIFIED	0	Speed units not defined
MS	1	Speed units in meters per second
KPH	2	Speed units in kilometres per hour

6.2.5.32 Registration message:TaskDefinition

The task definition field of a registration message shall conform to the values in **Table 49**.

NOTE *The task definition defines how a fusion node sends task messages to an edge node, and the valid values and data types the edge node can accept.*

Table 49 – "Task definition" fields

Type	Field name	Status	Protobuf Field Serial	Description
Int32	concurrent_tasks	M	1	The number of different tasks that can run at a time in this mode
RegionDefinition	region_definition	M	2	Several parameters to describe the region
Command	command	D List	3	Describes the command's name, type, completion time, units

6.2.5.33 Registration message:TaskDefinition–RegionDefinition

The region definition fields within the task definition field of a registration message shall conform to the values in **Table 50**.

NOTE *The region definition defines the region type(s) and coordinate system(s) in which an edge node can accept regions for a task.*

Table 50 – "Region definition" field within the task definition field

Type	Field name	Status	Protobuf Field Serial	Description
RegionType	region_type	M List	1	This is the type of region(s) the node can accept tasking for, for example, an active node like a radar might not be able to ignore certain areas
Duration	settle_time	D	2	Time to settle to normal performance in this mode
LocationType	region_area	M List	3	The type(s) of coordinate system in which the node can accept a region
ClassFilterDefinition	class_filter_definition	D List	4	The types of classes the node is capable of filtering on within the given mode
BehaviourFilterDefinition	behaviour_filter_definition	D List	5	The types of behaviours the node is capable of filtering on within a given mode

The region type field shall specify the type of region being defined in **Table 51**.

Table 51 – "Region type" field

Type	Enumeration value	Description
UNSPECIFIED	0	Region type not defined
AREA_OF_INTEREST	1	Used to define a region which is focused on by the edge node
IGNORE	2	Used to define a region which is ignored by the edge node
BOUNDARY	3	Used to define a region which in turn defines the boundary of the area of operations.
MOBILE_NODE_NO_GO_AREA	4	This defines a region that mobile nodes are not to enter
MOBILE_NODE_GO_AREA	5	This defines a region that mobile nodes are to remain within

6.2.5.34 Registration message:ClassFilterDefinition

This field shall be a list of parameters that the edge node shall filter on and shall conform to **Table 52**.

Table 52 – "Class filter definition" field within the range definition field

Type	Field name	Status	Protobuf Field Serial	Description
FilterParameter	filter_parameter	D List	1	—
SubClassFilterDefinition	sub_class_definition	D List	2	List of subclass filters
String	Type	M	3	Class type to filter on, for example, "Human", "Vehicle"

6.2.5.35 Registration message:ClassFilterDefinition-FilterParameter

The filter parameter field within the region definition field of a registration message shall conform to **Table 53**.

NOTE *This field is a list of class parameters the edge node can filter on. Typically, confidence is the main characteristic to filter on and the operator is usually "greater than".*

Table 53 – "Filter parameter" field within the region definition field

Type	Field name	Status	Protobuf Field Serial	Description
string	parameter	M	1	Name of the parameter that can be filtered
Operator enum	operators	M List	2	Operators that can be used during filtering

6.2.5.36 Registration message:ClassFilterDefinition-FilterParameter-Operator

The operators that can be used during filtering shall conform to the values in **Table 54**.

Table 54 – "Valid values" for operator enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Operator not defined
ALL	1	All
GREATER_THAN	2	Greater than (>)
LESS_THAN	3	Less than (<)
EQUAL	4	Equal to (=)

6.2.5.37 Registration message:SubClassFilterDefinition

The sub-class filter definition field within the region definition field of the registration message shall conform to **Table 55**.

Table 55 – "Sub-class filter" definition field within the region definition field

Type	Field name	Status	Protobuf Field Serial	Description
FilterParameter	filter_parameter	D List	1	—
Int32	level	M	2	Sub-class hierarchy level that this type is at; typically 1-3
String	type	M	3	Sub-class type to filter
SubClassFilterDefinition	sub_class_definition	D List	4	List of subclass filters

6.2.5.38 Registration message:BehaviourFilterDefinition

The behaviour filter definition field within the region definition field in a registration message shall conform to **Table 56**.

Table 56 – "Behaviour filter definition" field within the region definition field

Type	Field name	Status	Protobuf Field Serial	Description
FilterParameter	filter_parameter	D List	1	—
String	type	M	2	Behaviour type to filter on e.g. "Running"

Behaviour types shall conform to the behaviour taxonomy (see **7.4**).

6.2.5.39 Registration message:TaskDefinition

The command list of the task definition field of a registration message shall conform to **Table 57**.

NOTE 1 *The command list in the registration message gives the list of commands that an edge node supports, including the default commands that all edge nodes are to provide.*

Table 57 – "Command fields with the task definition" field

Type	Field name	Status	Protobuf Field Serial	Description
string	units	M	2	Valid values for this command e.g. Status
Duration	completion_time	M	3	For example, 1
CommandType enum	type	M	4	Type of task. Should match the type in **6.2.5.40**

NOTE 2 *The "name" field has been removed as it was a duplication of the "type" field.*

6.2.5.40 Registration message:TaskDefinition – Task commands

The list of commands that an edge node supports in a registration message shall conform to **Table 58**.

The commands shall be interpreted differently depending on whether the edge node is a sensor or an effector.

NOTE 1 *Further commands may be defined in the registration message.*

NOTE 2 *Tasking support for effectors specified in this BSI Flex is currently minimal and can be considered to be a prototype for more comprehensive functionality in the future.*

Table 58 – "Commands to be supported by an edge node" field

Type	Units	Protobuf Field Serial	Description for sensor edge node	Description for effector node
UNSPECIFIED	Not applicable	0	—	—
REQUEST	Registration	1	Request the node sends a registration message	
REQUEST	Reset	1	Request the node 'reboots' itself	
REQUEST	Status	1	Request the node sends a status report message	
REQUEST	Arm	1	Not used	Request the effector arms with the currently-selected mode(s), ready for action
REQUEST	Start	1	Request the sensor edge node starts sending detection and alert messages after a previous "Stop" request	Request the effector starts
REQUEST	Stop	1	Request the sensor edge node stops sending detection and alert messages until told to start again	Requests the effector stops
REQUEST	Take snapshot	1	Task a sensor edge node to take a snapshot. The URL to the snapshot should be provided in the "SensorTaskACK" message.	Not used
DETECTION_ THRESHOLD	Auto, Low, Medium, High	2	Allows the fusion node to adjust the sensor edge node's detection threshold or return it to its own "Auto" threshold	Not used
DETECTION_ REPORT_RATE	Auto, Low, Medium, High	3	Allows the fusion node to tell the sensor edge node to report more or less detections or return it to its own "Auto" rate	Not used
CLASSIFICATION_ THRESHOLD	Auto, Low, Medium, High	4	Allows the fusion node to tell the sensor edge node to adjust the sensitivity when classifying detections or return it to its own "Auto" rate	Not used

Table 58 – "Commands to be supported by an edge node" field *(continued)*

Type	Units	Protobuf Field Serial	Description for sensor edge node	Description for effector node
MODE_CHANGE	Default; others as defined in registration message e.g. 'Follow', comma-separated if multiple simultaneous modes required	5	Allows the fusion node to tell the node to change to a different operating mode(s)	
LOOK_AT	locationList, rangeBearingCone	6	Allows the fusion node to task a steerable node or a pointable platform to look at a particular location.	
MOVE_TO	LocationList	7	Allows a fusion node to task a mobile node to move to a new location. If LocationList contains more than one Location, this is interpreted as a series of waypoints. If LocationList contains just one Location, this is the final destination.	
PATROL	LocationList	8	A fusion node to task a mobile node to 'patrol' a route specified by the LocationList until a new command is received	
FOLLOW	FollowObject	9	Allows a fusion node to task a mobile or pointable node to follow another moving object	

6.2.5.41 Registration message:TaskDefinition – Task commands – Follow object

The "FollowObject" message shall be used to specify a platform node to follow a moving object. The object for the mobile node to follow shall be specified by the object_id field in the "Detection" message of the platform node.

NOTE 1 *It is expected that the platform node autonomously determines how it is to follow the moving object, for example, from how far back a mobile node is to follow the object. If a platform node does not have the capabilities to support the "Follow" command it does not declare this in its registration message. In that case, it might be possible for the fusion node to achieve a similar outcome by sending repeated 'Move_To' or 'Look_At' commands.*

Fields of the "FollowObject" message shall conform to **Table 59**.

Table 59 – "Follow Object" message field

Type	Field name	Status	Protobuf field serial	Description
string	follow_object_id	M	1	This is the object_id (ULID) of the object to follow, as specified in the "Detection" message from the mobile node dependent node that is tracking the object.

NOTE 2 *The "FollowObject" message supports the simple use case where only one target is to be followed. Guidance on multi-object use cases will be provided in later versions.*

6.3 Status message

6.3.1 Initial status message

As part of system initialization, an initial status report message shall be sent after the registration acknowledgement message has been received. This shall indicate the initial state.

NOTE 1 *Where possible, the physical location and field of view of the node should be included in status report messages.*

For sensor edge nodes providing detection messages in "RangeBearing" (spherical) coordinates, a node location shall be provided.

NOTE 2 *For sensor edge nodes that provide detection messages in cartesian coordinates, providing a node location is discretionary.*

NOTE 3 *External correction of node location may be used if the edge node does not have its own LAT/LONG or LAT/LONG accuracy is degraded.*

6.3.2 Status_Report message:StatusReport

6.3.2.1 General

All nodes shall produce regular status messages. The status message shall contain a time-stamp and the node's UUID.

NOTE 1 *This message is generally intended to be used as a 'heartbeat', providing basic confirmation of node function. However it may provide additional information including 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; and camera exposure time.*

NOTE 2 *In most applications a "Status" message, should be sent not less than once every 60 s and not more than once a second. The node defines how frequently it will send a status message in its registration message.*

If no status messages are received by a fusion node within three status intervals (defined in the edge node's registration message), the fusion node shall close the socket connection and wait for the edge node to re-establish communication.

A message handling application, if present in the system, shall also send regular status reports, to indicate it is still live and functioning and keep the channel open (in the case that no edge nodes have connected) (see **C.3.1**).

A regular status message shall conform to the values in **Table 60**.

Table 60 – Fields of a regular "Status report" message field

Type	Field name	Status	Protobuf Field Serial	Description
string	report_id	M	1	ULID of this report, incremented by the source
System enum	system	M	2	Overall status of sender
Info enum	Info	M	3	Unchanged, New. A flag to tell the fusion node if the information in this message is new or unchanged.
string	active_task_id	D	4	ID of the task being performed by the edge node
string	mode	M	5	Name of the mode the edge node is currently in, as defined in Registration message
Power	power	D	6	–
Location	node_location	D	7	–
LocationOrRangeBearing	field_of_view	D	8	–
LocationOrRangeBearing	coverage	D List	12	–
LocationOrRangeBearing	obscuration	D List	10	–
Status	status	D List	11	–

6.3.2.2 Status_Report message:StatusReport-System

This field shall specify the status of the node and shall conform to the values in **Table 61**.

Table 61 – Valid values for "Status report system" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	The status of the node is unknown
OK	1	The node is working normally
WARNING	2	Something has gone wrong with the node, but it can continue to operate
ERROR	3	There is an issue with the system
GOODBYE	5	The system is about to disconnect ***NOTE*** *Node sends a "GOODBYE" message to close the communication cleanly in case it needs to shut down and reboot.*

6.3.2.3 Status_Report message:StatusReport-Info

This field shall specify the type information contained in the status report and shall conform to the following values in **Table 62**.

Table 62 – Valid values for "Info" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Info not defined
NEW	1	New information is available in this status report
UNCHANGED	2	No new information is contained in this status report

6.3.2.4 Status_Report message:StatusReport-Mode

This field shall be included where an edge node provides multiple modes or states. The field value shall conform to the list of values registered by the node in its registration message.

NOTE 1 *When changing between nodes there can be short periods where the node does not function in its usual manner as it adapts to the new mode. This period is reported by the 'settle_time' in* ***Table 27****. Other factors can also affect the performance of a node (see* ***6.3.2.9****).*

NOTE 2 *Normally the edge node works in default mode, but the fusion node can request it changes into any other mode specified in the node registration message.*

6.3.2.5 Status_Report message:StatusReport-Power

This field shall be used to report the status of the edge node power supply if available. This field shall conform to the values in **Table 63**.

Table 63 – "Power" field of a status message

Type	Field name	Status	Protobuf Field Serial	Description
int32	level	D	3	Battery level, as a percentage, e.g. 50
PowerSource enum	source	D	4	The current power source being used by the node (see **Table 64**)
PowerStatus enum	status	D	5	OK, Fault (see **Table 65**)

The level field shall be an integer representing the level of the remaining battery as a percentage of the total with valid values from 0 to 100.

Table 64 – Valid values for the "Power source" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	The power source is unset
OTHER	1	The power source is of a different type from those listed in this table
MAINS	2	The power source is the mains
INTERNAL_BATTERY	3	The power source is a battery contained within the node
EXTERNAL_BATTERY	4	The power source is a battery external to the node
GENERATOR	5	The power source is a generator
SOLAR_PV	6	The power source is a solar photovoltaic cell
WIND_TURBINE	7	The power source is a wind turbine
FUEL_CELL	8	The power source is a fuel cell

Table 65 – Valid values for the "Power status" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	The power status is not specified
OK	1	The power source is operating normally
FAULT	2	There is a fault with the power source

6.3.2.6 Status_Report message:StatusReport-Node_Location

Location shall be provided as a location message (see **5.2**) with discretionary fields for altitude and error.

NOTE *Node location is a discretionary field that reports the current geographical location of the edge node. With a fixed edge node, this may be reported once during initialization, although typically it is reported in every status report to aid debugging.*

6.3.2.7 Status_Report message:StatusReport-Field_of_view

Field of view shall be reported either as a range "bearing cone" message (see **5.2.3**) or a location list proto message. For a location list, points shall be in order around the boundary of the field of view polygon.

If a field of view changes, the info enumeration (**Table 62**) shall be set to "NEW" and the new field of view reported. An info enumeration of "NEW" with no reported field of view shall be interpreted as the sensor having lost view.

NOTE *Field of view is a discretionary field that reports the current field of view of the edge node. With a fixed edge node, this should be reported once during initialization and then updated on change. For steerable edge nodes, this may be reported regularly. In either case, the registered reporting interval (see* ***Table 21****) should be followed.*

6.3.2.8 Status_Report message:StatusReport-Coverage

Coverage shall be reported either as a range bearing cone message or a location list message. For a location list, points shall be in order around the boundary of the coverage polygon.

NOTE *Coverage is a discretionary field that reports the extent of the area a node (either edge node or fusion node) can cover but not necessarily all at once. Typically, this can only be reported for scanning or steerable edge nodes. For example, a pan tilt zoom camera may have 360° coverage but only a field of view of 45°.*

6.3.2.9 Status_Report message:StatusReport-Obscuration

Obscuration shall be reported either as a range bearing cone message or a location list message. For a location list message, points shall be in order around the boundary of the obscuration polygon.

NOTE *Obscuration is a discretionary field that can define one or more regions that the sensor edge node 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. To determine what region is obscured behind a moving target, the fusion node can calculate the region using the reported object size and location in the detection report.*

6.3.2.10 Status_Report message:StatusReport-Status

This field shall conform to values in **Table 66**.

When a sensor edge node is moving and not able to carry out sensing tasks while doing so, it shall report as transition. If an edge node has received a stop request command, it shall report "stopped".

NOTE *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.*

Table 66 – Fields of the "status" element of a status message field

Type	Field name	Status	Protobuf Field Serial	Description
StatusLevel enum	status_level	D	1	Indicates the severity of the condition
StatusType enum	status_type	M	4	Type of information being reported, e.g. weather
string	status_value	D	3	Value of report (see **6.2.5.9**)

6.3.2.11 Status_Report message:StatusReport-StatusLevel

This field shall specify the severity of the status condition being reported and shall conform to the values in **Table 67**.

Table 67 – Valid values for "Status level" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Status level not defined
INFORMATION_STATUS	2	Conditions have been detected that might affect other nodes
WARNING_STATUS	3	Performance is degraded
ERROR_STATUS	4	Something is severely affecting capability

NOTE *Enumeration value 1 has been deprecated.*

6.3.2.12 Status_Report message:Status-Status_type

The "Status_type" field shall have one of the values in **Table 68**.

Table 68 – Valid values for "Status_type" field

Type	Enumeration value	Description
UNSPECIFIED	0	No status type is specified
INTERNAL_FAULT	1	There is a fault internal to the node
EXTERNAL FAULT	2	There is a fault external to the node
ILLUMINATION	3	A report about illumination levels
WEATHER	4	A report about weather conditions
CLUTTER	5	A report about clutter
EXPOSURE	6	A report about exposure
MOTIONSENSITIVITY	7	A report about motion sensitivity
PTZ_STATUS	8	A report about the status of a PTZ node
PD	9	Probability of detection
FAR	10	False alarm rate
NOT_DETECTING	11	The node is not detecting
PLATFORM	12	A report about a platform node
OTHER	13	—

The status type and status value fields shall be consistent with the status, type and unit fields defined in the registration message.

6.3.3 One-off status message

One-off status messages shall not be used.

NOTE *Historically, previous versions of SAPIENT used "one-off" status messages for alert reporting. This is now deprecated in favour of alert messages (see* ***6.6****).*

6.4 Detection message

6.4.1 Detection_Report message:DetectionReport

Detection messages shall be sent by a node to indicate detection, classification or behaviour of an object. This message shall report each detection (or, optionally, a group of detections or track data).

NOTE 1 *"Track data" is a discretionary element used to provide the fusion node with more information about the movement of an object, where available.*

The detection message shall contain the node's UUID, a timestamp. The detection message shall include a unique ID number for the object in the form of a ULID, and real-world location of the detected object.

NOTE 2 *The actual content of the messages can potentially be different for each sensor edge node, depending on its capabilities. No individual node can be expected to fill in all the possible fields.*

Where sensor edge node performance allows, classification information shall also be included for each detected object.

Each sensor edge node shall provide all information described in its registration message.

The field values of the detection message shall conform to the values given in **Table 69**.

Table 69 – Field values of the "Detection report" field

Type	Field name	SAPIENT field status	Protobuf field serial	Description
String	report_id	M	1	Unique identifier (ULID) of this message. The date/time component of the ID is updated with each report.
String	object_id	M	2	Unique identifier (ULID) of object detected. If fusion node is not capable of tracking or identifying an object, then it can match the "reportID".
string	task_id	D	3	Identifier of task that the sensor edge node was carrying out when this detection occurred.
string	state	D	4	Whether special case such as object lost
oneof	location_oneof	M	5,6	Either "Location" (see **5.2.2**) or "RangeBearing" (see **5.2.2.5**) to be provided to give the location of the detection
float	detection_confidence	D	7	Probability that the sensor edge node has detected a real object value between 0 and 1; 0 is low confidence, 1 is certain.
TrackObjectInfo	track_info	D List	8	Additional metadata about the track
PredictedLocation	prediction_location	D	9	Prediction of where the object will be at a later time
TrackObjectInfo	object_info	D List	10	Additional metadata about the detected object
DetectionReport Classification	classification	D List	11	List of possible types of object this could be, e.g. human, vehicle. Can also describe what the probability of each class is.

Table 69 – Field values of the "Detection report" field *(continued)*

Type	Field name	SAPIENT field status	Protobuf field serial	Description
DetectionReport Behaviour	behaviour	D List	12	List of possible activities this object could be performing, e.g. walking, running. Also, what the probability of each behaviour is.
AssociatedFile	associated_file	D List	13	List of URLs to associated media such as image snapshots, 3D point clouds.
Signal	signal	D List	14	List of signals detected as part of this detection.
AssociatedDetection	associated_detection	D List	15	In a fused detection, the list of detections that were used to generate this detection
DerivedDetection	derived_detection	D List	16	A list of fused detections derived from this detection
oneof	velocity_oneof	D	19	Velocity of detected object in ENUVelocityUnits (see **5.3.2**)
String	colour	D	22	Colour of object
String	id	D	23	ID of object

NOTE 3 *In a hierarchical architecture a RangeBearing alone is not valid, therefore the low-level Fusion node should perform a conversion from RangeBearing to Location.*

6.4.2 Detection message detail

6.4.2.1 Report identification

Report ID shall be a ULID of the detection report message. The date/time component of the field shall be updated with each report.

6.4.2.2 Object identification

Object ID shall be a ULID generated by a sensor edge node that uniquely identifies an object. If a sensor edge node is not capable of associating multiple detections of the same object then the object ID generated shall be unique to each detection message.

6.4.2.3 Task identification

Task Identifier shall be the "task_id" that the sensor edge node was carrying out when this detection occurred.

NOTE *Future versions of BSI Flex 335 may reconsider the validity of the concept of a default task.*

6.4.2.4 State

This is a discretionary field; valid values shall be free text or null.

__NOTE__ Local conventions can make agreed use of this field. For example, for camera-based sensor edge nodes, when an object remains static in the field of view for a long time it can merge into the background. If detected as such, the state field should be set to "lost."

6.4.2.5 Location

The location field shall be either a location (see **5.2.2 Location message**:Location) or "RangeBearing" (see **5.2.2.5 Range_Bearing message**:RangeBearing) to give the location of the detection.

6.4.2.6 Detection confidence

The detection confidence field shall be set to the probability that the node has detected a real object, i.e. a value between 0 and 1, where 0 is low confidence and 1 is certain.

6.4.2.7 Track_Info

The "trackInfo" field is deprecated because the "TrackObjectInfo" shall be used.

6.4.2.8 Detection_Report message:PredictedLocation

The field values of the predicted location field shall conform to **Table 70**.

Table 70 – Field values of the "Predicted location" field

Type	Field name	Status	Protobuf Field Serial	Description
PredictedLocation	predicted_location_oneof	M	1,2	Either "Location" (see **5.2.2**) or "RangeBearing" (see **5.2.2.5**) shall be provided to give the location of the predicted detection
google.protobuf. Timestamp	predicted_timestamp	D	3	Timestamp of predicted detection in UTC if different from detection report timestamp

6.4.2.9 Detection_Report message:TrackObjectInfo

The field values of the object Information field shall conform to **Table 71**.

Table 71 – Field values of the "Track object info" field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Name value pairs of metadata associated with track
string	value	M	2	The value of the object info being reported
Float	error	D	3	Error value associated with value

6.4.2.10 Detection message – colour

The detection colour field shall be set to a text colour description.

6.4.2.11 Detection message – identifier

The identifier field shall be set to a unique identifier of the individual object detected, if known.

***NOTE** For example this could be the serial number.*

6.4.2.12 Detection message – classification

The field values of the "DetectionReportClassification" field shall conform to the values in **Table 72**.

Table 72 – Field values of the "Detection report classification" field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Object type as defined in Clause **7**, e.g. human, vehicle. The format of this string is <namespace>:<value>.
float	confidence	D	2	Probability that the object is of this object type (class)
SubClass	sub_class	D List	3	List of possible sub classes of object that this object could be (see **6.4.2.13**). ***NOTE** If the object subtype has been defined in a taxonomy extension (see **6.2.5.27**) then possible extended subclasses are listed here.*

***NOTE** <namespace> is now a mandatory extension to the type string in this version of BSI Flex 335. This defaults to "sapient_core" (see **7.2**) if the taxonomy has not been extended.*

6.4.2.13 Detection_Report message:subclass

The field values of the "subclass" field shall conform to the values in **Table 73**.

Table 73 – Field values of the "Subclass" field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Object type as defined in Clause **7**, e.g. human, vehicle. The format of this string shall be <namespace>:<value>.
float	confidence	D	2	Probability that the object is of this object type (class)
int32	level	M	3	Level of the sub class
SubClass	sub_class	D List	4	List of possible sub classes of object that this object could be. ***NOTE** If the object subtype has been defined in a taxonomy extension (see **6.2.5.27**) then possible extended subclasses are listed here.*

***NOTE** <namespace> is now a mandatory extension to the type string in this version of BSI Flex 335. This defaults to "sapient_core" (see **7.2**) if the taxonomy has not been extended.*

6.4.2.14 Detection_Report message:Behaviour

The field values of the behaviour field shall conform to **Table 74.**

Table 74 – Field values of the "Behaviour" field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Object activity as defined in **7.4** e.g. walking, running
float	confidence	D	2	Probability that the object is undertaking each activity

6.4.2.15 Detection message – "AssociatedFile"

The field values of the "Associated File" field shall conform to **Table 75**.

Table 75 – Field values of the "Associated file" field

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Type of file, e.g. image
string	url	M	2	URL to the media (**C.4.1**). Typically, the node identifier is included as a sub-folder in the URL.

***NOTE** See Note to **5.4** regarding file.*

6.4.2.16 Detection_Report message:Signal

The field values of the Signal field shall conform to **Table 76**.

***NOTE** For all fields in this table, units should be included in "Registration:Detection" definition (see **6.2.5.16**).*

Table 76 – Field values of the "Signal" field

Type	Field name	Status	Protobuf Field Serial	Description
float	Amplitude	M	1	Signal amplitude, typically in dB
float	start_ frequency	D	2	Minimum frequency of signal, typically in MHz.
float	centre_ frequency	M	3	Centre frequency of signal, typically in MHz.
float	stop_ frequency	D	4	Maximum frequency of signal, typically in MHz.
float	pulse_ duration	D	5	Duration of signal pulse, typically in ns.

6.4.2.17 Detection message – associated detection

This field shall specify the associated detection. This shall use the common associated detection message (see **5.5**).

6.4.2.18 Detection message – associated detection – association type

This field shall specify the relationship to the associated detection. This shall use the common associated detection – "AssociatedRelation" enumeration (see **5.5.2**).

6.4.2.19 Detection message – derived detection

The derived detection field shall be used by a fusion node to indicate when a fused detection is generated. This field shall be used to add a reference to the fused detection from the record of the original detection.

***NOTE** When used with "AssociatedDetection", this provides two linked lists linking original and derived, fused detections.*

The field values of the "derived_detection" field shall conform to **Table 77**.

Table 77 – Field values of the "Derived detection" field

Type	Field name	Status	Protobuf Field Serial	Description
google.protobuf.Timestamp	Timestamp	M	1	Timestamp of associated detection
string	node_id	M	2	Unique identifier of fusion node that was the source of the detection
string	object_id	M	3	Unique identifier of the detection

6.5 Task message

6.5.1 General description

Task messages shall be sent by the fusion node via the message handling application to task individual nodes.

***NOTE 1** The overall philosophy used is that commands and filtering options are task-based with a discretionary associated region defined in the task command. The particular message information is very much dependent on the capabilities of the individual node, as described in the "Registration" message.*

The general form of a task message shall be request ID, node ID and the task definition, which includes the following sections:

a) (discretionary) a region definition related to the task; and

b) (discretionary) a command or action related to the task.

At least one of a) or b) shall be defined. Where b) is not defined and the node has a default task, the node shall execute its default task. Where the node does not have a default task, b) shall be defined.

***NOTE 2** Examples of tasks and their uses are:*

a) in region X look for threats of type Y;

b) ignore all detections in region Z;

c) look at location (X, Y); [this could be used to cross-cue (via the fusion node) from a detection from a sensor edge node to another edge node];

d) request a snapshot; and

e) the sensor edge node's false alarm rate is unacceptably high [or P(d) is unacceptably low]. In this case the thresholds should be adjusted accordingly.

***NOTE 3** Command based tasks tend to be short lived or instantaneous whereas region tasks tend to persist until another task is received.*

6.5.2 Task message:Task

The field values of the task message shall conform to the values in **Table 78**.

Table 78 – Field values of the "Task" message

Type	Field name	Status	Protobuf Field Serial	Description
String	task_id	M	1	—
String	task_name	D	2	Optional name of task
String	task_description	D	3	Optional description of task
google.protobuf. Timestamp	task_start_time	D	4	UTC time to start task
google.protobuf. Timestamp	task_end_time	D	5	UTC time to end task
Control enum	control	M	6	"Start", "Stop", "Pause", "Default"
Region	region	D List	7	Optional list of regions associated with the task
Command	command	D	8	Populated if this task is a command

6.5.3 Task ID

Task messages shall include a task ID.

NOTE 1 *The task ID is a ULID generated by the fusion node to uniquely identify a task so that the acknowledgement message and any subsequent detection report messages have a reference.*

NOTE 2 *A task ID field that is blank indicates the default task to be undertaken when there is no other task allocated to the node.*

6.5.4 Task message:Control

The control field in the task message shall define what action to perform on that task and shall conform to the following values in **Table 79**.

Table 79 – Valid values for "Control" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	No control defined
START	1	Initializes and schedules or starts the task or restarts a paused task
STOP	2	Stop the task, remove its definition and revert to previous task
PAUSE	3	Stop the task but keep it defined for later restart, revert to previous task

NOTE *A stop message does not require a 'region' or 'command' section.*

When a task completes or is stopped by a stop or pause value, the node shall revert to any previously defined task. If a node does not have a previously defined task, it shall stop executing any task until it receives a new task.

6.5.5 Task message:Region

*COMMENTARY ON **6.5.5***

Region tasks are designed to control the geographical area of operation within the coverage of steerable nodes or field of view of fixed nodes. Typically by default, sensor edge nodes detect from their full field of view and so a region task would limit the area to report from.

6.5.5.1 General

The "field values of region" shall conform to **Table 80**.

Table 80 – Field values of "Region"

Type	Field name	Status	Protobuf Field Serial	Description
RegionType enum	type	M	1	Area of interest, ignore, boundary
string	region_id	M	2	Region ID number generated by the fusion node
string	region_name	M	3	For example, car park
LocationOrRangeBearing	region_area	M	4	Either reported as a "RangeBearingCone"message (see **5.2.3**) or a 'LocationList' message (see **5.2.2**)
ClassFilter	class_filter	D List	5	List of class filters to apply to region. If omitted, then no filtering is applied
BehaviourFilter	behaviour_filter	D List	6	List of behaviour filters to apply to region. If omitted, then no filtering is applied

If the fusion node intends to remove regions from the default task, then a default task message shall be sent with only the remaining regions.

6.5.5.2 Task message:Region-ClassFilter

The field values of the "ClassFilter" shall conform to **Table 81**.

Table 81 – Field values of "Class filter"

Type	Field name	Status	Protobuf Field Serial	Description
parameter	parameter	M	1	Typically filter on confidence value
string	type	M	2	Type of classification to filter on
SubClassFilter	sub_class_filter	D List	3	Subclasses to filter on
DiscreteThreshold enum	priority	D	4	Priority of this filter

NOTE *If supported by the sensor edge node, object type can be used to restrict the types of objects that can be reported.*

6.5.5.3 Task message:Region-SubClassFilter

The field values of the "SubClassFilter" shall conform to **Table 82**.

Table 82 – Field values of "Subclass filter"

Type	Field name	Status	Protobuf Field Serial	Description
parameter	parameter	M	1	Typically filter on confidence value
string	type	M	2	Class to filter on
SubClassFilter	sub_class_filter	D List	3	List of any child sub classes to filter on
DiscreteThreshold enum	priority	D	4	Priority of this filter

6.5.5.4 Task message:Region-DiscreteThreshold

The "DiscreteThreshold" enumeration field shall specify the threshold in discrete steps. It shall define priorities and shall conform to the values in **Table 83**.

Table 83 – Valid values for "Discrete threshold" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Not set
LOW	1	—
MEDIUM	2	—
HIGH	3	—

6.5.5.5 Task message:Region-BehaviourFilter

The field values of the "Behaviour Filter" field shall conform to the values in **Table 84**.

Table 84 – Field values of "Behaviour filter"

Type	Field name	Status	Protobuf Field Serial	Description
parameter	parameter	M	1	Typically filter on confidence value
string	type	D	2	Behaviour to filter on
DiscreteThreshold enum	priority	D	3	Priority of this filter

6.5.6 Task message:Command

6.5.6.1 General

The command part of the message shall be populated for command-based tasks. Only one command message shall be issued at a time and so only one of the top-level fields shall be populated.

The field values of the command field shall conform to **Table 85**.

Table 85 – Field values of "command"

Type	Field name	Status	Protobuf Field Serial	Description
oneof	command	M	1,2,3,4,5,6	One of the following types of command selected and a (discretionary) parameter may be supplied
string	request	—	1	The request being asked for
DiscreteThreshold enum	detection_threshold	—	2	The requested sensitivity of the sensor edge node for the task
DiscreteThreshold enum	detection_report_rate	—	3	The requested reporting rate of the sensor edge node for the task
DiscreteThreshold enum	classification_threshold	—	4	The requested classification sensitivity of the sensor edge node for the task
string	mode_change	—	5	Mode change for the edge node

Table 85 – Field values of "command" *(continued)*

Type	Field name	Status	Protobuf Field Serial	Description
LocationOrRangeBearing	look_at	—	6	"LookAt" command for pointing sensor edge nodes
string	command_parameter	D	8	Optional parameter string for some commands
LocationList	move_to	—	9	"Move_To" command for instructing a mobile node to move to a new location
LocationList	patrol	-—	10	"Patrol" command for instructing a mobile node to continually patrol a route until instructed otherwise
FollowObject	follow	—	11	"Follow" command for instructing a mobile node to follow a moving object until instructed otherwise

6.5.6.2 Task message:Command – request

The "request" command with a parameter of registration shall allow the fusion node to request the edge node to send the registration message again.

The "request" command with a parameter of status shall allow the fusion node to request a full status report message with the current value of all fields supported by that edge node.

The "request" command with a parameter of reset shall allow the fusion node to stop the edge node's current task and return to its default state.

The "request" command with a parameter of stop shall allow the fusion node to request that the sensor edge node stop sending all detection report messages. The sensor edge node shall continue sending status report messages to maintain the connection to the data agent.

The "request" command with a parameter of start shall make the sensor edge node resume sending detection report messages after a "request – stop" command.

6.5.6.3 Task message:Command – detection_threshold

*COMMENTARY ON **6.5.6.3***

The individual sensor edge nodes are expected to exhibit a level of autonomy. Therefore, the fusion node is not expected to use a "long screwdriver" to set low-level parameters within an individual sensor edge node. The fusion node may, however, choose to request that a sensor edge node adjust its internal thresholds to reduce its False Alarm Rate (FAR), or increase its probability of detection [P(d)], based for example on a comparison of the detection performance of one ASM with another with an overlapping field of view.

The command "detection_threshold" shall allow the fusion node to adjust the sensor edge node's detection threshold. The "classification_threshold" field shall be of type "DiscreteThreshold". Values shall conform to the enumeration specified in **6.5.5.4**.

6.5.6.4 Task message:Command – detection_report_rate

NOTE *The fusion node might wish to request an individual sensor edge node to alter the frequency at which it reports data, for example to throttle back the detections if they were supplied too frequently.*

The command "detection_report_rate" shall allow the fusion node to adjust the sensor edge node's report rate. The "classification_threshold" field shall be of type "DiscreteThreshold". Valid values shall conform to the enumeration specified in **6.5.5.4**.

6.5.6.5 Task message:Command – classification_threshold

NOTE *The fusion node may choose to request an individual sensor edge node to alter the sensitivity at which it classifies data, for example to allow for "looser" searches in challenging environments.*

The command "classification_threshold" shall allow the fusion node to raise or lower the classification threshold. The classification_threshold field shall be of type "DiscreteThreshold". Values shall conform to the enumeration specified in **6.5.5.4.**

6.5.6.6 Task message:Command – mode_change

The "mode_change" message shall force the edge node to change to one of its operating modes defined in the registration message (see **Table 49**). Valid values shall be those defined in the registration message.

If an edge node only supports one mode, then this command shall be ignored.

6.5.6.7 Task message:Command – look_at

The "look_at" command shall task the sensor edge node to suspend its current tasking, look at a specific location or range bearing (see **5.2**) to see if it can detect anything, and then return to its ongoing tasking.

The sensor edge node shall return a detection message if it detects an object at that location.

NOTE *A task ack with a complete status implies that nothing was found if there were no detection reports.*

The sensor edge node shall determine the most appropriate field of view to use in response to the "look_at" command. If this is not the default field of view, then this shall be reported by a status message to accompany any detection message response to the "lookAt" command.

6.5.6.8 Task message:Command – move_to

The "move_to" command shall task a mobile node to cease any existing "move_to", "patrol", or "follow" commands it is executing and move to the location specified in the most recent "move_to" command.

If the "LocationList" specified with the "move_to" command contains more than one location, this shall be interpreted as a list of waypoints to be followed on route to the final destination (the last entry in the "LocationList"). If the "LocationList" contains only one location, this shall be interpreted as the final destination.

The mobile node shall autonomously determine the route it needs to take to reach the final destination, or between waypoints, whilst remaining within the technical and environmental limitations of the mobile node.

On reaching the final destination, the mobile node shall cease moving until new "move_to", "patrol", or "follow" commands are received.

6.5.6.9 Task message:Command – patrol

The patrol command shall task a mobile node to cease any existing "move_to", "patrol", or "follow" commands it is executing and to begin patrolling the route specified in the most recent patrol command. The mobile node shall continuously patrol the specified route until a new "move_to", "patrol", or "follow" command is received.

If the first and last entries in the LocationList are identical, the LocationList shall be interpreted as defining the outline of a completed polygon to be patrolled by the mobile node. If the first and last entries of the LocationList are not identical, on reaching the last entry in the "LocationList" the mobile node shall retrace its steps back to the start.

6.5.6.10 Task message:Command – follow

The follow command shall task a mobile node to cease any existing "move_to", "patrol", or "follow" commands it is executing and to begin following the object specified in the most recent "follow" command. The mobile node shall continue to follow the object until a new "move_to", "patrol", or "follow" command is received.

The "follow" command accepts a list of (ULID) "object_ids" of the object to be followed. The "object_ids" are specified by any sensor node(s) mounted on the mobile node.

6.5.7 Task message protocol

The fusion node shall send task messages to lower nodes.

Any task shall override the default task. If a second override task is received and the node cannot support multiple tasks then it shall report that in the acknowledgement message when tasked with the second task.

The node shall respond to the task message with one or more "TaskACK" acknowledgement messages during the lifetime of the task.

The acknowledgement message shall state whether the node accepts or rejects the task.

NOTE 1 *It is valid to include additional data such as a URL to a snapshot.*

If a node cannot conform to a control message, it shall send a "TaskACK" acknowledgement message where the status field is set to the value "rejected".

NOTE 2 *A description of the reason for rejection may be included in the reason field of the message.*

6.5.8 Sensor/effector modes

A node shall operate in one of the modes defined in the "Registration" message, with the node changing between modes, as a state machine, with changes initiated by the mode change request.

NOTE 1 *A sensor/effector mode is defined as a distinct way of operating.*

NOTE 2 *A non-steerable edge node may operate in a single mode. A steerable edge node is likely to have multiple modes that define how it responds to tasking from the fusion node. 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.*

NOTE 3 *An effector should typically have a number of effecting modes that are specific to its effector type. For example, a jamming effector might be capable of operating at a number of frequencies, with different waveforms or using antennas with different gains.*

NOTE 4 *A set of predefined modes may be used to avoid each edge node defining modes independently. This approach may be application-specific, local conventions can make agreed use of this feature. To conform to a predefined mode set, the edge node does not need to support all the modes in the set.*

NOTE 5 Any effector has a potential safety implication. A "soft" effector which causes a UAV to fall to ground could cause harm to humans or property, and a "hard" effector, i.e. using bullets/shells, etc., directed at ground, air or sea targets has even more safety implications. Therefore, any system which includes effectors should be subject to an approved Safety Case.

The modes supported by a node shall be declared in the registration message.

6.5.9 Task acknowledgement message

6.5.9.1 Task_ack message:TaskAck

Task acknowledgment messages shall complete the handshaking and allow a node to accept or reject a task. The task acknowledgement message shall be used if a task is rejected for an operational reason; however, if the received task message contains an error, a SAPIENT "Error" message shall be sent and the task acknowledgement shall not be sent.

The field values of the acknowledgement message shall conform to values in **Table 86**.

Table 86 – Field values of "Task acknowledgement" message

Type	Field name	Status	Protobuf Field Serial	Description
string	task_id	M	1	Task ID the acknowledgement is in response to
TaskStatus enum	task_status	M	2	The node's response to the tasking request
string	reason	D List	5	The reason(s) the task was rejected or failed if the task was rejected or failed
AssociatedFile	associated_file	D	4	File associated with task being acknowledged

6.5.9.2 Task_ack message:TaskStatus

This field shall specify the status of the task and shall conform to the values in **Table 87**.

Table 87 – Valid values for "Task status" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	The task status has not been defined
ACCEPTED	1	The requested task has been accepted and will be carried out
REJECTED	2	The requested task has been rejected by the edge node
COMPLETED	3	The requested task has already been completed
FAILED	4	The edge node has previously accepted a task, but has been unable to successfully complete the task

6.6 Alert message

6.6.1 Alert message:Alert

Alert messages shall be used for two use cases:

a) for a node to transmit information to the GUI; and

b) for a node to transmit information to the fusion node.

NOTE *Alert conditions are not defined in this BSI Flex; they are system-specific. The term "alert condition" refers to whatever conditions the node raises alerts on. Typical examples of alert conditions are: for use case a), a fusion node flags a fused detection report to the GUI; and for use case b), an edge node reports a non-detection event that is not covered in a status or detection report.*

If an alert condition is true, the node shall send an alert message. Alert messages shall conform to **Table 88**.

Table 88 – "Alert" message field

Type	Field name	Status	Protobuf Field Serial	Description
string	alert_id	M	1	Unique identifier of alert (ULID)
AlertType enum	alert_type	D	2	Type of alert; "Information" implies no response required
AlertStatus enum	status	D	3	Current status of alert: "Active", "Acknowledge", "Reject", "Ignore", "Clear"
string	description	D	4	Text description of alert
LocationOrRangeBearing	location	D	5,6	Providing a Cartesian location is optional and should only be done if relevant to do so. Only a location or a range bearing should be provided.
string	region_id	D	7	Optional field to refer to an existing region of interest as designated by the fusion node (ULID)
DiscretePriority enum	priority	D	8	"Low", "Medium" "High"
float	ranking	D	9	0.0 – 1.0

Table 88 – "Alert" message field *(continued)*

Type	Field name	Status	Protobuf Field Serial	Description
float	confidence	D	10	0.0 – 1.0
AssociatedFile	associated_file	D List	11	List of associated data files
AssociatedDetection	associated_detection	D List	12	List of associated detection points
string	additional_information	D	13	Any additional information for the fusion node

6.6.2 Alert ID

The Alert ID field shall be set to a ULID for the alert.

6.6.3 Alert message:AlertType

The "AlertType" field shall specify the severity of alert being reported and shall conform to the values in **Table 89**.

Table 89 – Valid values for "Alert type" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Alert type is undefined
INFORMATION	1	This alert is for information only
WARNING	2	This alert shows something that may need a human to review it
CRITICAL	3	This node has a serious problem and might fail soon
ERROR	4	The node has a problem but is still operating
FATAL	5	The node has failed
MODE_CHANGE	6	The edge node notifies the fusion node that it intends to autonomously change mode

6.6.4 Alert message:AlertStatus

The alert status field shall specify the current state of the alert being reported and shall conform to the values in **Table 90**.

Table 90 – Valid values for "Alert status" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Alert status is not defined
ACTIVE	1	Alert is active and has yet to be responded to
ACKNOWLEDGE	2	Alert has been previously acknowledged
REJECT	3	Alert has been previously rejected
IGNORE	4	Alert has been previously ignored
CLEAR	5	Enables a previously acknowledged alert to be cleared

6.6.5 Description

The description field shall contain a free text description of the alert.

6.6.6 Location

The location field shall be set to a location (see **5.2**) for the alert. "One of" location or range bearing shall be used.

NOTE *Providing a location is discretionary and should only be done if relevant.*

6.6.7 Region ID

The "Region ID" field is discretionary, but if set shall refer to an existing region of interest as designated by the fusion node (ULID).

6.6.8 Priority

The discrete priority field shall specify the priority in discrete steps. It shall conform to the values in **Table 91**.

Table 91 – Valid values for "Priority" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Discrete priority not defined
LOW	1	Low priority set
MEDIUM	2	Medium priority set
HIGH	3	High priority set

6.6.9 Ranking

This field shall be a float, bounded by the values of 0 and 1.

6.6.10 Confidence

This field shall be a float, bounded by the values of 0 and 1.

6.6.11 Associated file

The Associated file field of the alert message shall conform to **Table 92**.

Table 92 – "Associated file" field of the alert message

Type	Field name	Status	Protobuf Field Serial	Description
string	type	M	1	Type of file, e.g. image
string	url	M	2	URL to the media; typically the node identifier is included as a sub folder in the URL

***NOTE** See Note in **5.4** regarding file.*

6.6.12 Associated detection

The associated detection field of an alert message shall conform to **Table 13**.

6.6.13 Alert_ack message:AlertAck

Where the alert message is used by the node to transmit information to the fusion node, the "alert acknowledgement" message shall be used by the fusion node to acknowledge to the node that it has received the "alert" message. If the "alert" message contains an error, a SAPIENT "Error" message shall be returned and an "alert acknowledgement" message shall not be sent. The "alert acknowledgement" message shall conform to **Table 93**.

Table 93 – "Alert acknowledgement" message field

Type	Field name	Status	Protobuf Field Serial	Description
string	alert_id	M	1	Alert ID the acknowledgement is in response to
string	reason	D List	4	The reason(s) the task was rejected if the task was rejected
AlertAckStatus enum	alert_ack_status	M	5	The node's response to the alert acknowledgement

The status field of the alert acknowledgement message shall specify the status of the alert and shall conform to the values in **Table 94**.

NOTE 1 *"AlertStatus" (Field serial 2) is deprecated. "AlertAckStatus" should be used.*

Table 94 – Valid values for "Alert Acknowledgement status" enumeration field

Type	Enumeration value	Description
UNSPECIFIED	0	Alert status is not defined
ACCEPTED	1	Alert is active and has yet to be responded to
REJECTED	2	The requested alert has been rejected
CANCELLED	3	The requested alert has already been cancelled

NOTE 2 *In the case of **6.6.1** (node transmitting information to the fusion node; see **6.5.1**), node-generated alert messages can provide a means for a node to request something that needs a response from the fusion node. An example is where a node intends to autonomously change mode but requires permission from the fusion node.*

Where the alert message is used by the edge node to get a response from the fusion node and the fusion node rejects the request, the message shall set the value in the status field to reject. The "reason" field shall be used to allow the fusion node to inform the edge node when it can try to send this request again. To allow the edge node to send this request again retry shall be used in the reason field, followed by the time interval and time units. An empty reason field shall be used to indicate that this request should not be sent again.

NOTE 2 *An example of this could be "retry 10 seconds."*

6.7 Error message

If an invalid message is received by the message handling application from either an edge node or fusion node, then the message handling application shall respond by sending an error message back to the source. This shall contain the time of the message, the content of the invalid message and a message describing the error and the fields given in **Table 95**.

Table 95 – Fields of an "Error" message

Type	Field name	Status	Protobuf Field Serial	Description
bytes	packet	M	1	The packet or part of the message which caused the error
string	error_message	M List	3	Description of the error(s) being reported

7 Taxonomy

***NOTE** A taxonomy defines a set of descriptive nouns in a hierarchy with three levels including object classes, behaviours and other characteristics.*

7.1 General

A sensor edge node shall provide the lowest level of classification that is within its capabilities.
The namespace of the SAPIENT core taxonomy shall be "sapient_core".

7.2 SAPIENT Core Object classification taxonomy

Messages that require the use of the SAPIENT core object classification taxonomy shall conform to **Table 96**.

***NOTE 1** The "unknown" top-level class specified in BSI Flex 335 v1.0 has been deprecated in favour of 'Other' in this version.*

"Other" shall be reserved for where a sensor edge node is certain an object detected belongs to a class not in the core taxonomy, rather than when sensor edge node is not able to distinguish between one of the known classes in the core taxonomy. Where a sensor edge node computes that an object could belong to several possible classes, it shall distribute probability amongst those classes. The sum of all probabilities shall equal 1.

***NOTE 2** For example, if an object is definitely a human or a vehicle but it is unclear which, then the sensor edge node should report human: 0.5, vehicle: 0.5, rather than other: 1.0.*

7.3 Taxonomy extensions

*COMMENTARY ON **7.3***

Many sensor edge nodes have native capability to detect and distinguish objects and behaviour that are not part of the SAPIENT core taxonomy. The taxonomy in BSI Flex 335 v2.0 still retains the three-tier hierarchy of: Class, SubClass, SubSubClass from BSI Flex 335 v1.0, but has been expanded to include an approach to allow edge nodes to declare the taxonomy extensions they will use at registration time and then use them at detection time.

Taxonomy extensions shall dock into existing classes in the SAPIENT core taxonomy, either specified classes or other.

***NOTE** The following examples are expanded in **Annex B**:*

- *Example 1: A sensor edge node is able to detect and distinguish satellites. Satellite classes would dock into the top level (Level 0) "Other" class. This is a local namespace.*
- *Example 2: A sensor edge node is able to detect and distinguish e-Bikes. The Automotive Ontology (AUTO) has the class MotorizedBicycle (https://auto.schema.org/MotorizedBicycle) (as distinct from Motorbike) which is to be used. MotorizedBicycle would dock into the Level 2 class Land Vehicle – 2 Wheels – Other.*
- *Example 3: A sensor edge node is able to detect and further classify large commercial shipping into Tanker Ships. This class would dock in to the Level 2 class Sea Vessel – Large Ship – Commercial. This is derived from the DTIC Thesaurus https://discover.dtic.mil/thesaurus/.*

*See **Figure 7** for a graphical illustration of the docking positions in these examples.*

Figure 7 – Example taxonomy extensions docked in to the SAPIENT core taxonomy

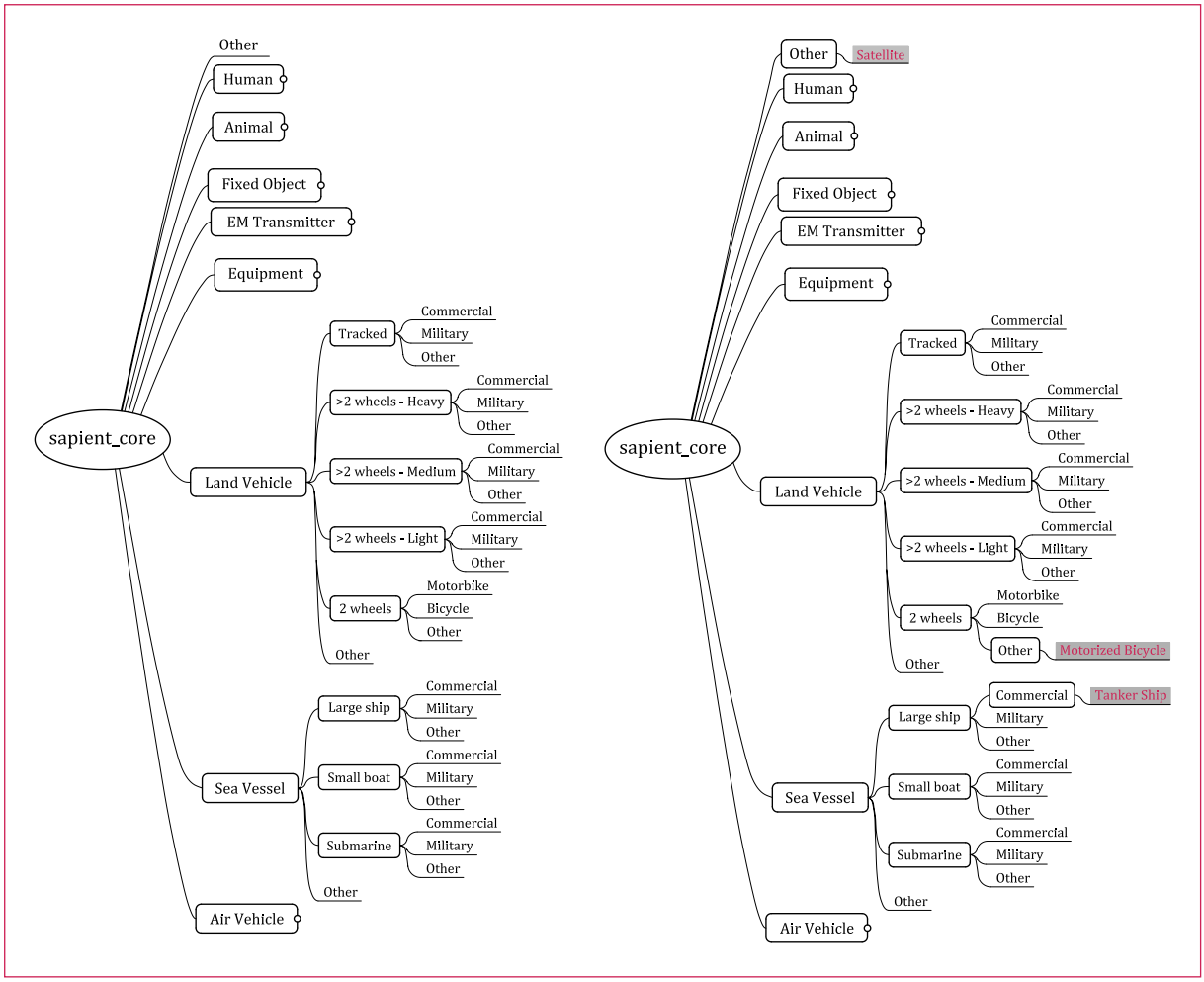


Table 96 – Full list of class names in the SAPIENT object classification taxonomy

Class	Subclass Level 1	Subclass Level 2
Other	—	—
Human	Male	—
	Female	—
Animal	Bird	—
	Horse	—
	Other	—
Fixed object	Building	—
	Fixed tower/mast	—
	Stationary Rotator	—
	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	—

Table 96 – Full list of class names in the SAPIENT object classification taxonomy *(continued)*

Class	Subclass Level 1	Subclass Level 2
Equipment	Weapon	Gun
		Missile launcher
		Missile
		Bomb
		IED/mine
		Grenade
		Other
	Passive sensor	Camera
		Acoustic
		Other
	Other	—
Land vehicle	Tracked	Commercial
		Military
		Other
	>2 wheels - Heavy	Commercial
		Militar
		Other
	>2 wheels - Medium	Commercial
		Military
		Other
	>2 wheels - Light	Commercial
		Military
		Other
	2 wheels	Motorbike
		Bicycle
		Other
	Other	—

Table 96 – Full list of class names in the SAPIENT object classification taxonomy *(continued)*

Class	Subclass Level 1	Subclass Level 2
Sea vessel	Large ship	Commercial
		Military
		Other
	Small boat	Commercial
		Military
		Other
	Submarine	Commercial
		Military
		Other
	Other	—
Air vehicle	Manned rotary wing	Commercial
		Military Other
	Manned fixed wing	Commercial
		Military Other
	UAV rotary wing	Commercial
		Military Other
	UAV fixed wing	Commercial
		Military Other
	Other	—

7.4 Object behaviour taxonomy

*COMMENTARY ON **7.4***

Object behaviour is reported independently of object type using behaviour types. Any confidence values associated with the behaviour can relate to the object class type with the highest confidence, e.g. if an object is detected as a 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.

Messages that require the use of the object behaviour taxonomy shall conform to **Table 97**.

***NOTE** The list in **Table 97** is likely to be extended in future releases of BSI Flex 335 as new behaviours are identified. Not all behaviours will necessarily be supported on a given system. This is implementation-specific.*

Table 97 – Valid values for "Object behaviour taxonomy" field

Type	Description
Walking	Subject is walking
Running	Subject is running
Crawling	Subject is crawling
Climbing	Subject is climbing
Digging	Subject is digging
Throwing	Subject is throwing something
Loitering	Subject is loitering
Active	Object is active (moving or operating)
Passive	Object is passive (not moving or operating)
Other	—

Annex A (informative)

List of significant changes from BSI Flex 355 v1.0

A.1 Clause 4

Updated descriptions of new concepts of Platforms and Hierarchical architecture. Figure 1 – new behaviour for message handling application.

a) Table 1 – added new field to allow users to add additional information as required.

A.2 Clause 5

a) Table 4 – New subclause after Table 4 specifies that altitude is reported as metres above location datum

b) Table 11 – east_rate and north_rate ENU Parameters made Mandatory

c) Table 13 – timestamp made Conditional

A.3 Clause 6

a) Table 15 – one new element added – config_data

b) Table 16 – “ack_response_reason” made a list

c) Table 18 – three new node types – mobile, pointable, fusion

d) Table 22 – “Days” added

e) Table 23 – Zone made conditional

f) Table 26 – new status values

g) Table 27 – “DetectionDefinition” made Conditional

h) New Table 43 – “Taxonomy dock definition field for taxonomy extensions”

i) New Table 44 – “Subclass definition for taxonomy extensions”

j) New Table 51 – “Region Type” and more descriptions added

k) Table 52 – two new Region types added

l) Table 53 – The field parameter is no longer list, whereas operators has now been made a list - this reflects the associated protobuf file.

m) Table 58 – Arm is not applicable to sensors – table changed

n) Table 58 – Three new Command Types added

o) New Table 59 – “Follow Object” message

p) Table 60 – “mode” field made Mandatory, “coverage” now a list

q) New Table 68 – “Valid values for Status_Type field” (replaces a set of bullets)

r) Table 68 – added value “Other” of protobuf value 13

s) Table 77 – Timestamp made Mandatory

t) New Table 83 – “Valid values for DiscreteThreshold enumeration field” (replaces a set of bullets)

u) Table 86 – “reason” now a list

v) Table 93 - changed “AlertStatus” to “AlertAckStatus” – Field name ‘alert_ack_status’ value 5

w) Table 95 – “error_message” now a list to allow a list of reasons

A.4 Clause 7

a) New Figure – Example taxonomy extensions docked in to the sapient_core taxonomy (Figure 7)

b) Table 96 – Addition of “Other” in Taxonomy

c) New Table 97 – “Valid values for Object Behaviour Taxonomy” (replaces a set of bullets)

A.5 Summary of changes made to Annex B

a) Mandatory fields “node_definition” and “config_data” added to registration message example

b) Mandatory field “acceptance” added to registration ack message example

c) Mandatory fields “info and mode” added to status message example

d) Mandatory fields “detection_location” added to detection report example

e) Added “alert”, “alert ack” and “error” message examples

A.6 Summary of changes made to Protobuf files

A.6.1 sapient_message.proto

added field “additional_information”

A.6.2 range_bearing.proto

added new message “LocationOrRangeBearing”

A.6.3 registration.proto

In enum TimeUnits, added “Days”

In message “Registration”– new fields added

“dependant_nodes”

“reporting_region”

“configuration_data”

In enum “NodeType”, new options

Mobile_Node

Pointable_Node

Fusion_Node

In enum “ModeType” – one new option added:

MODE_TYPE_DEFAULT

In enum DetectionReport Category – one new option added:

DETECTION_REPORT_CATEGORY_SIGNAL

In message “DetectionClassDefinition” – one new field added:

taxonomy_dock_definition

In message “Command”

Removed “name” field because it is a duplicate

In enum “CommandType” – three new options added:

COMMAND_TYPE_MOVE_TO

COMMAND_TYPE_PATROL

COMMAND_TYPE_FOLLOW

In enum “RegionType” - two new options added

REGION_TYPE_MOBILE_NODE_NO_GO_AREA

REGION_TYPE_MOBILE_NODE_GO_AREA

New messages added:

TaxonomyDockDefinition

ExtensionSubclass

ConfigurationData

A.6.4 registration_ack.proto

ack_response_reason now “repeated” to allow for a list of reasons

A.6.5 task.proto

In message “Command” – three new fields added:

move_to

patrol

follow

removed message “LocationOrRangeBearing” as it is now declared in range_bearing.proto

A.6.6 task_ack.proto

In message TaskAck – the reason field is now “repeated” to allow a list of reasons

A.6.7 alert_ack.proto

In message AlertAck – the reason field is now “repeated” to allow a list of reasons

“alert_status” renamed to “alert_ack_status”

A.6.8 proto_options.proto

Additional verification options added – not directly as a result of BSI v2.0 update

A.6.9 error.proto

In message “Error” – the “error_message” field is now “repeated” to allow a list of error messages

New Protobuf file – follow.proto

Defines the “FollowObject” message, used in “registration.proto” and “task.proto”

Annex B (informative)
Example messages

B.1 General

This annex contains examples of common messages. They show a camera-based sensor edge node registering, and sending a status report and detection report. The sensor edge node can then be tasked to complete a "LookAt" command by the fusion node. Messages here are serialized into JSON for readability. In operation the messages are serialized into binary by protobuf.

B.2 Registration message

An example registration message is shown in **Figure B.1**.

Figure B.1 – Example registration message


```
{
  "timestamp": "2022-11-03T10:05:06.141204500Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "registration": {
    "node_definition": [
      {
        "nodeType": "CAMERA",
      }
    ],
    "ICD_version":"BSI Flex 335 v2.0",
    "name": "MyCamera sensor edge node",
    "capabilities": [
      {
        "category": "Test",
        "type": "Camera",
        "units": "Url"
      }
    ],
    "statusDefinition": {
      "statusInterval": {
        "units": "SECONDS",
        "value": 5
      },
      "locationDefinition": {
        "locationUnits": "LATLNGDEGM",
        "locationDatum": "WGS84E"
      },
      "statusReport": [
       {
          "category": "SENSOR",
          "type": "SensorLocation",
          "onChange": true
       },
       {
```

Figure B.1 – Example registration message *(continued)*


```
            "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
                      }
                    ]
```

Figure B.1 – Example registration message *(continued)*

```
                }
              ]
            }
          "TaxonomyDockDefinition": [           {
              "Dock_class_namespace": "sapient_core",
              "Dock_class": "Other"
              "Extension_subclass": [
                {
                  "Subclass_namespace": "Sat_catalogue1",
                  "Subclass_name": "satellite",
                  "units": "probability",
                }
              ]
            }
          "TaxonomyDockDefinition": [
            {
              "Dock_class_namespace": "sapient_core",
              "Dock_class": "Land Vehicle.2 Wheels.Other"
              "Extension_subclass": [
                {
                  "Subclass_namespace": " https://auto.schema.org/",
                  "Subclass_name": "MotorizedBicycle",
                  "units": "probability",
                }
              ]
            }
          "TaxonomyDockDefinition": [
            {
              "Dock_class_namespace": "sapient_core",
              "Dock_class": "Sea Vessel.Large Ship.Commercial"
              "Extension_subclass": [
                {
                  "Subclass_namespace": " https://discover.dtic.
                  mil/thesaurus/",
                  "Subclass_name": "Tanker Ships",
                  "units": "probability",
                }
              ]
            }
          ]
        },
        "task": [
          {
            "regionDefinition": {
              "regionType": [
                "IGNORE"
              ],
```

Figure B.1 – Example registration message *(continued)*

```
                "settleTime": {
                  "units": "SECONDS",
                  "value": 1
                },
                "regionArea": [
                  {
                    "locationUnits": "LATLNGDEGM"
                  }
                ]
              },
              "command": [
                {
                  "name": "REQUEST",
                  "units": "Location",
                  "completionTime": {
                    "units": "SECONDS",
                    "value": 1
                  }
                }
              ],
              "type": "REQUEST"
              "ConfigurationData": [
                {
                  "manufacturer": "Indt Optics",
                  "model": "A1",
                }
              ],
            }
          ]
        }
      ]
    }
```

B.3 Registration ack message

An example registration ack message is shown in **Figure B.2**.

Figure B.2 – Example "registration ack" message

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

B.4 Status report message

An example status report message is shown in **Figure B.3**.

Figure B.3 – Example of a "status report" message


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


NOTE *The following examples illustrate typical values with which a status message can be populated. The first illustrates an example where rain has been detected that does not affect this sensor edge node but might affect other edge nodes. The second illustrates an example where the performance of the edge node has been degraded by external conditions (e.g. rain or snow), but is still able to perform its current tasking. The third illustrates an example where the edge node has been degraded by external conditions (e.g. illumination), so it is unable to perform its current tasking.*

Example information status values to indicate rain has been detected

status_level	INFORMATION_STATUS
status_type	Weather
status_value	Rain

Example warning status values to indicate performance is degraded

status_level	WARNING_STATUS
status_type	ExternalFault
status_value	n/a

Example error status values to indicate performance has been degraded

status_level	ERROR_STATUS
status_type	ExternalFault
status_value	Illumination

B.5 Detection report message

An example detection report message is shown in **Figure B.4**.

Figure B.4 – Example 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
      }
    ]
    "detection_location" {
      "x": 51.17273763593,
      "y": -1.82237324543,
      "coordinateSystem": "LATLNGDEGM",
      "datum": "WGS84E"
  }
}
```

B.6 Task message

An example task message is shown in **Figure B.5**.

Figure B.5 – example of a 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"
            }
          ]
        }
      }
    }
  }
}
```


B.7 Task ACK message

An example task ack message is shown in **Figure B.6**.

Figure B.6 – Example 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"
  }
}
```

An example alert message is shown in **Figure B.7**.

Figure B.7 – Example alert message

```
{
  "timestamp": "2022-11-03T10:34:07.819389200Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "alert": {
    "alert_id": "139bf8c7-84da-4c00-a068-4d58dca747b8",
  }
}
```

An example alert ack message is shown in **Figure B.8**.

Figure B.8 – Example alert ack message

```
{
  "timestamp": "2022-11-03T10:34:011.819189200Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "alertack": {
    "alert_id": "139bf8c7-84da-4c00-a068-4d58dca747b8",
    "alert_ack_status": "UNSPECIFIED",
  }
}
```

An example error message is shown in **Figure B.9**.

Figure B.9 – Example error message

```
{
  "timestamp": "2022-11-03T10:40:011.819189200Z",
  "nodeId": "b902eb31-97f1-4acf-b994-da3ec901074e",
  "error": {
    "packet": "detection_location" {
      "x": 51.17273763593,
      "y": -1.82237324543,
      "coordinateSystem": "(•⌣•)",
      "datum": "WGS84E",
    }
    "error_message": {
     "invalid coordinateSystem",
    }
  }
}
```

Annex C (informative) Additional information

C.1 Guidance on external interfaces

In order that SAPIENT can be integrated, it has to permit connections to external systems to meet some use cases. Examples of where SAPIENT has been connected to other systems are shown on https://www.gov.uk/guidance/sapient-autonomous-sensor-system.

C.2 Additional resources

This BSI Flex is accompanied by .proto files that are effectively the machine-readable version of this BSI Flex. These can be accessed by visiting https://www.gov.uk/guidance/sapient-autonomous-sensor-system. Adopters and users of BSI Flex 335 are encouraged to download and use these .proto files to guarantee correct operation between SAPIENT components. A user who, instead, generates their own .proto files for their component cannot guarantee their component will work with other SAPIENT components using different .proto files.

Prospective adopters and users of BSI Flex 335 can access additional resources to assist with establishing conformity to the standard. These include a basic test harness that enables test messages to be sent to a SAPIENT component being tested, and to test messages received from the component under test to confirm that they conform to the SAPIENT standard. Details of how to obtain the test harness and .proto files can be found at www.gov.uk/sapient.

C.3 Message handling application description

C.3.1 Common functionality

Some SAPIENT systems optionally use a message handling application, often referred to as middleware, to implement functionality not covered by this BSI Flex. If incorporated, the message handling application is essentially 'transparent' to message content and node behaviour (**0.4** SAPIENT architecture).

The message handling application has its own registration, i.e. it registers with the fusion node as a node in its own right. The message handling application shall register with no modes and no detection reports. In operation it sends regular status reports to keep the channel open (in the case that no edge nodes have connected yet).

The following examples of message handling application functionality are given as guidance:

NOTE *These are optional and not specified by BSI Flex 335.*

a) provide a mechanism to transfer data to and from the fusion node, including:
   1) forwarding acknowledgement messages; and
   2) forwarding the task message to its destination;
b) provide a mechanism to persist data, independent of any data store in the fusion node;
c) provide a GUI and mechanism to transfer data to and from it; and

d) provide additional validation of messages beyond what exists in the fusion node; in this case, the message handling application may:

   1) validate the mandatory fields in the message: if the message is invalid, the information it contains should not be forwarded (this is unlikely to impede any validation functions the fusion node performs); and

   2) generate an error message for invalid messages (see **6.7** Error message).

If message handling applications have a UUID "node_id", it should be included in the message wrapper (see **Table 1**). In that case "destination_ids" should be replaced, as appropriate, as the message passes through the middleware. This does not affect the general principle of 'transparency'.

C.3.2 Example middleware

Example middleware might become available in the future. There are plans for this to be referenced in future versions of this BSI Flex and be made available from https://www.gov.uk/guidance/sapient-autonomous-sensor-system.

C.4 Interface for raw data

Although the philosophy within this BSI Flex is that fusion node functionality is based on the BSI Flex 335 messages produced by the edge nodes, there are occasions where it is necessary to stream raw data from a sensor. Although this is not part of the BSI Flex 335 specification, this subclause provides guidance on the functionality to facilitate this.

C.4.1 File data

BSI Flex 335 standardizes the mechanism for responding to a request for file data but does not standardize where the file data is stored. Local agreement is required to define shared file services accessible by all nodes that are to respond to such requests and appropriate file storage location should be set in the configuration file of the Node.

Upon request for file data, the node responds with a URL to the file location, as specified in **5.4**.

C.4.2 Streaming video

Streaming video is not supported as part of this BSI Flex. However, for many use cases this is a key requirement, so this section provides guidance on how this can be implemented.

Streaming video may conform to either of the following main forms:

a) compressed video streams over Ethernet (Video over IP); or

b) direct digital video signals (e.g. HDMI, HD-SDI).

These two forms are discussed further in **C.4.2.1** and **C.4.2.2.** Sensor equipment may internally use industrial and/or scientific cameras using protocols such as GigE Vision or USB3 Vision or specialist connectivity such as "CameraLink" or "Coaxpress". Any non-supported forms may be converted to one of the two supported forms in a) and b) above. Conversion of such video may be achieved by the supplier using an intermediary PC or laptop. The computer can display the video via an HDMI connection. Ideally, the computer display output can provide a representation of the imagery at full resolution.

C.4.2.1 Compressed video data

Network video cameras provide compressed video bitstreams in a multitude of different compression standards and container formats. Network video can be encoded using one of the main standards listed below:

a) Motion JPEG (MJPEG);

b) MPEG-4;

c) H.264 (MPEG-4 AVC); or

d) H.265 (HEVC).

This guidance does not support frame rates that exceed standard high-definition video formats (i.e. 60 frames per second and 1 920×1 080 pixels). Video captured progressively (as opposed to interlaced) is preferred.

All network video streams can be readily ingested into VLC (https://www.videolan.org/) using commonly used streaming containers such as MPEG-TS or MP4/MOV and communications protocols such as RTSP or UDP.

Minimal latency when generating video is ideal. This might require the use of less efficient video compression standards (e.g. MJPEG) or require limiting the capability of more advanced compression standards (e.g. avoiding the use of bidirectional prediction).

Networked video streams may use variable bitrate encoding but should encode video at a constant frame rate. Bitrates to achieve satisfactory quality encoding in most circumstances are shown in **Table C.1**.

Table C.1 – Bitrates to achieve satisfactory quality encoding

	Image (Intra-frame) compression (MJPEG)	Video Compression (MPEG4 AVC, H26x)
Standard Definition	10 Mbit/s	1-2 Mbit/s
High-Definition	50-60 Mbit/s	5-12 Mbit/s

Networked video streams can be presented to the rest of the system using wired RJ45 connections.

C.4.2.2 Raw video data

Raw video can be output in 1080p30, 1080p25, 720p60 and 720p50 formats, using either HDMI or HD-SDI connectivity. Computer monitor outputs such as Display Port or DVI can readily be converted to HDMI using appropriate adapters.

HDMI connections can be achieved using standard full-size HDMI connection. HD-SDI can be connected using standard BNC connectors. Progressive video is preferred.

Annex D (informative)
Taxonomy extensions

The following are potentially useful resources for taxonomy extensions (this is not an exhaustive list).

- Defense Technical Information Center (DTIC) Thesaurus provides a broad, multidisciplinary subject-term vocabulary that aids information search and retrieval. Subject terms, called Descriptors, are organized into hierarchies, where series of narrower terms are linked to broader terms. The DTIC Thesaurus is not copyrighted. No license is needed to use it.
- Taxonomy NATO C3 https://www.nato.int/cps/en/natohq/topics_157573.htm
- NATO – AINTP-03 (RESTRICTED) THE NATO MILITARY INTELLIGENCE DATA EXCHANGE STANDARD

Developer libraries and other useful resources are available at:

- Web Ontology Language (OWL) https://www.w3.org/OWL/
- SKOS Simple Knowledge Organization System https://www.w3.org/2004/02/skos/

Bibliography

Other publications

Useful websites

Defence Science and Technology Library, SAPIENT autonomous sensor system
https://www.gov.uk/guidance/sapient-autonomous-sensor-system

Protocol Buffers Documentation
https://developers.google.com/protocol-buffers

Requirements for Internet Hosts
https://www.rfc-editor.org/rfc/rfc1122

A Universally Unique IDentifier (UUID) URN Namespace
https://datatracker.ietf.org/doc/html/rfc4122

Universally Unique Lexicographically Sortable Identifier
https://github.com/ulid/spec

NCI Agency holds NATO's live-testing counter-drone exercise
https://www.ncia.nato.int/about-us/newsroom/nci-agency-holds-natos-livetesting-counterdrone-exercise.html

World Geodetic System (WGS84)
https://gisgeography.com/wgs84-world-geodetic-system/

This page is deliberately left blank.

This page is deliberately left blank.

This page is deliberately left blank.

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards-based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email cservices@bsigroup.com.

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Relations	**Tel:** +44 345 086 9001	**Email:** cservices@bsigroup.com
Subscription Support	**Tel:** +44 345 086 9001	**Email:** subscription.support@bsigroup.com
Knowledge Centre	**Tel:** +44 20 8996 7004	**Email:** knowledgecentre@bsigroup.com
Copyright & Licensing	**Tel:** +44 20 8996 7070	**Email:** copyright@bsigroup.com

BSI, 389 Chiswick High Road
London W4 4AL, United Kingdom
www.bsigroup.com