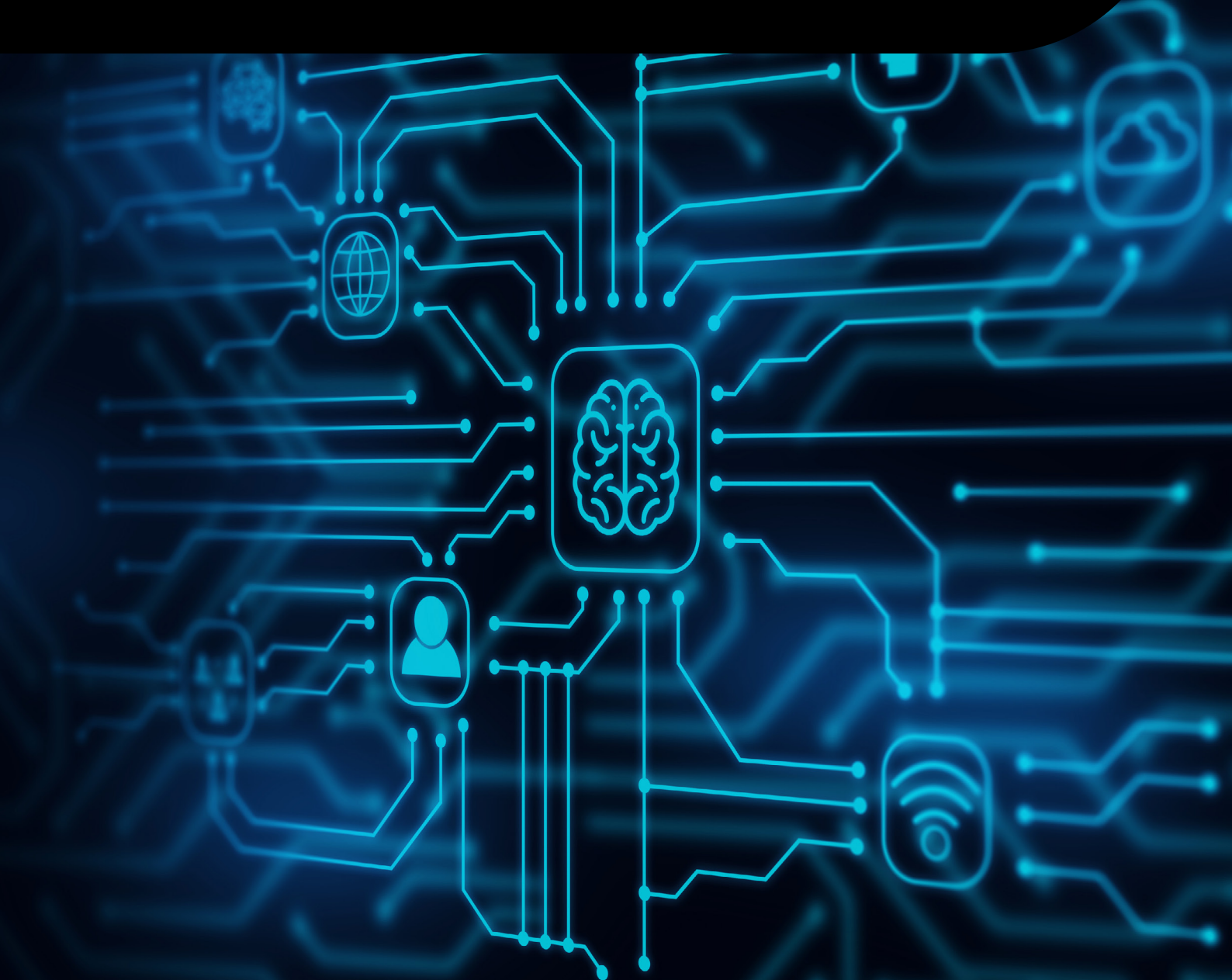


SAPIENT Network of autonomous sensors and effectors – Interface control document – Specification

Version 2 March 2024



BSI Flex 335 v2.0:2024-03

Publishing and copyright information

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

© The British Standards Institution 2024
Published by BSI Standards Limited 2024

ISBN 978 0 539 26777 8

ICS 35.030, 35.080, 95.040

No copying without BSI permission except as permitted by copyright law.

Release History

First Version July 2023
Second Version March 2024

Contents

Foreword v

0 Introduction vii

0.1 General vii

0.2 SAPIENT concept vii

0.3 Applications of SAPIENT viii

0.4 SAPIENT architecture viii

1 Scope 1

2 Normative references 2

3 Terms, definitions and abbreviated terms 3

3.1 Terms and definitions 3

3.2 Abbreviated terms 5

4 Interface description 6

4.1 General 6

4.2 Message framing/structure 7

4.3 Version-number string 8

4.4 Initialization 8

4.5 Normal operation 8

4.6 Node alerts during normal operations 9

4.7 Task messages 9

4.8 Node shutdown 9

4.9 Lost connection 10

4.10 Platforms 10

4.11 Hierarchical architecture 12

5 Inner Message Types 14

5.1 General 14

5.2 Location types 14

5.3 Velocity 21

5.4 Associated_File message:AssociatedFile 22

5.5 Associated_detection message:AssociatedDetection message 23

6 Outer message types 24

6.1 General 24

6.2 Registration 24

6.3 Status message 51

6.4 Detection message 57

6.5 Task message 64

6.6 Alert message 73

6.7 Error message 77

7 Taxonomy 78

7.1 General 78

7.2 SAPIENT Core Object classification taxonomy 78

7.3 Taxonomy extensions 78

7.4 Object behaviour taxonomy 83

Annexes

Annex A (informative)	84
A.1 Clause 4.....	84
A.2 Clause 5.....	84
A.3 Clause 6.....	84
A.4 Clause 7.....	85
A.5 Summary of changes made to Annex B.....	85
A.6 Summary of changes made to Protobuf files.....	85
Annex B (informative)	
Example messages	87
B.1 General.....	87
B.2 Registration message.....	87
B.3 Registration ack message.....	90
B.4 Status report message.....	91
B.5 Detection report message.....	92
B.6 Task message.....	93
B.7 Task ACK message.....	93
Annex C (informative)	
Additional information	95
C.1 Guidance on external interfaces.....	95
C.2 Additional resources.....	95
C.3 Message handling application description.....	95
C.4 Interface for raw data.....	96
Annex D (informative)	
Taxonomy extensions	98
Bibliography	99
Other publications.....	99
Useful websites.....	99
List of figures	
Figure 1 – SAPIENT architecture.....	ix
Figure 2 – Typical SAPIENT message sequence.....	x
Figure 3 – Message sequence for a platform architecture.....	11
Figure 4 – An example of a complex platform node.....	11
Figure 5 – Message sequence for a hierarchical architecture.....	13
Figure 6 – East, North, Up (ENU) frame of reference.....	21
Figure 7 – Example taxonomy extensions docked in to the SAPIENT core taxonomy.....	79
Figure B.1 – Example registration message.....	87
Figure B.2 – Example “registration ack” message.....	90
Figure B.3 – Example of a “status report” message.....	91
Figure B.4 – Example detection report message.....	92
Figure B.5 – example of a task message.....	93
Figure B.6 – Example task ACK message.....	93
Figure B.7 – Example alert message.....	94
Figure B.8 – Example alert ack message.....	94
Figure B.9 – Example error message.....	94

List of tables

Table 1 – Overall message wrapper structure.....	7
Table 2 – Coordinate system for different types of location information	15
Table 3 – “Location” message information	16
Table 4 – Valid values for the location coordinate system enumeration field	17
Table 5 – Valid values for the location datum enumeration field.....	17
Table 6 – “LocationList” message	17
Table 7 – Information for the “RangeBearing” message field.....	18
Table 8 – Message field information for the “RangeBearingCone” message.....	19
Table 9 – Valid values for the “RangeBearingCoordinateSystem enum” field	20
Table 10 – Valid values for the “RangeBearingDatum enum” field	20
Table 11 – Information for the “ENUVelocity” field	22
Table 12 – “Associated file” message field information.....	22
Table 13 – “Associated detection” message field information	23
Table 14 – Valid values for the “Associated relation” enumeration field	23
Table 15 – General structure of a registration message.....	25
Table 16 – “Registration acknowledgement” message format.....	26
Table 17 – “NodeDefinition” message.....	27
Table 18 – Node type enumeration	27
Table 19 – “Capability” field types	28
Table 20 – “Status definition” fields	29
Table 21 – “Duration” fields	29
Table 22 – Units of time	30
Table 23 – “Location” type fields	30
Table 24 – Status report fields	31
Table 25 – “Status report” category.....	31
Table 26 – Status report – status values	32
Table 27 – “Mode definition” fields	33
Table 28 – Valid values for the “ModeType” enumeration field.....	34
Table 29 – Valid values for “ScanType” enumeration field.....	34
Table 30 – Valid values for “TrackingType” enumeration field	35
Table 31 – “Mode parameter” field	36
Table 32 – Registration – “ConfigurationData” field	36
Table 33 – “Detection definition” field	37
Table 34 – “Detection definition geometric error” field.....	38
Table 35 – “Detection report” field	38
Table 36 – Valid values for “Detection report category” enumeration field	39
Table 37 – Detection report – “Track object info” value	39
Table 38 – “Detection class definition” field	40
Table 39 – Valid values for “confidence definition enumeration” field.....	40
Table 40 – “Definition detection performance” value field.....	41
Table 41 – “Detection definition – class definition” field.....	41
Table 42 – “Sub-class” definition of the detection definition field	42
Table 43 – “Taxonomy dock” definition field for taxonomy extensions	42
Table 44 – “Subclass definition for taxonomy extension” field	43
Table 45 – “Behaviour” definition field	43
Table 46 – “Velocity type” field.....	44
Table 47 – “ENU Velocity units” field.....	44
Table 48 – “Speed units” field.....	44
Table 49 – “Task definition” fields	45
Table 50 – “Region definition” field within the task definition field	45
Table 51 – “Region type” field	46
Table 52 – “Class filter definition” field within the range definition field	46

Table 53 – “Filter parameter” field within the region definition field	47
Table 54 – “Valid values” for operator enumeration field	47
Table 55 – “Sub-class filter” definition field within the region definition field	47
Table 56 – “Behaviour filter definition” field within the region definition field	48
Table 57 – “Command fields with the task definition” field	48
Table 58 – “Commands to be supported by an edge node” field	49
Table 59 – “Follow Object” message field	51
Table 60 – Fields of a regular “Status report” message field	52
Table 61 – Valid values for “Status report system” enumeration field	53
Table 62 – Valid values for “Info” enumeration field	53
Table 63 – “Power” field of a status message	54
Table 64 – Valid values for the “Power source” enumeration field	54
Table 65 – Valid values for the “Power status” enumeration field	55
Table 66 – Fields of the “status” element of a status message field	56
Table 67 – Valid values for “Status level” enumeration field	56
Table 68 – Valid values for “Status_type” field	57
Table 69 – Field values of the “Detection report” field	58
Table 70 – Field values of the “Predicted location” field	60
Table 71 – Field values of the “Track object info” field	61
Table 72 – Field values of the “Detection report classification” field	61
Table 73 – Field values of the “Subclass” field	62
Table 74 – Field values of the “Behaviour” field	62
Table 75 – Field values of the “Associated file” field	63
Table 76 – Field values of the “Signal” field	63
Table 77 – Field values of the “Derived detection” field	64
Table 78 – Field values of the “Task” message	65
Table 79 – Valid values for “Control” enumeration field	65
Table 80 – Field values of “Region”	66
Table 81 – Field values of “Class filter”	67
Table 82 – Field values of “Subclass filter”	67
Table 83 – Valid values for “Discrete threshold” enumeration field	67
Table 84 – Field values of “Behaviour filter”	68
Table 85 – Field values of “command”	68
Table 86 – Field values of “Task acknowledgement” message	72
Table 87 – Valid values for “Task status” enumeration field	72
Table 88 – “Alert” message field	73
Table 89 – Valid values for “Alert type” enumeration field	74
Table 90 – Valid values for “Alert status” enumeration field	75
Table 91 – Valid values for “Priority” enumeration field	75
Table 92 – “Associated file” field of the alert message	76
Table 93 – “Alert acknowledgement” message field	76
Table 94 – Valid values for “Alert Acknowledgement status” enumeration field	77
Table 95 – Fields of an “Error” message	77
Table 96 – Full list of class names in the SAPIENT object classification taxonomy	80
Table 97 – Valid values for “Object behaviour taxonomy” field	83
Table C.1 – Bitrates to achieve satisfactory quality encoding	97

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

The British Standards Institution retains ownership and copyright of this BSI Flex. BSI Standards Limited, as the publisher of the BSI Flex, reserves the right to withdraw or amend this BSI Flex on receipt of authoritative advice that it is appropriate to do so.

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.

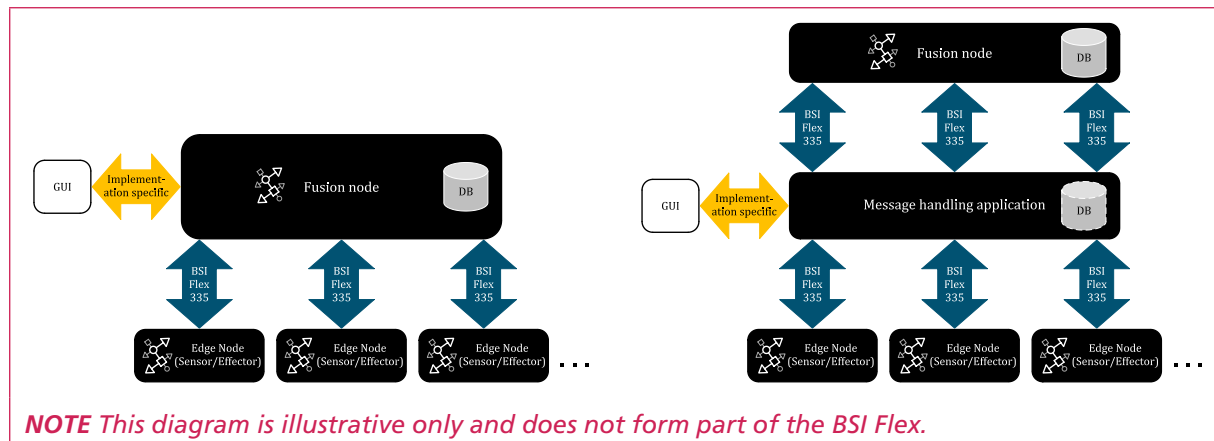
The SAPIENT concept shows benefits in situations where no single sensor has all the information required for situational awareness so fusion of the information from multiple sensors is required. Additionally, since SAPIENT explicitly provides for node management, the concept shows increased benefit where 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

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:

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

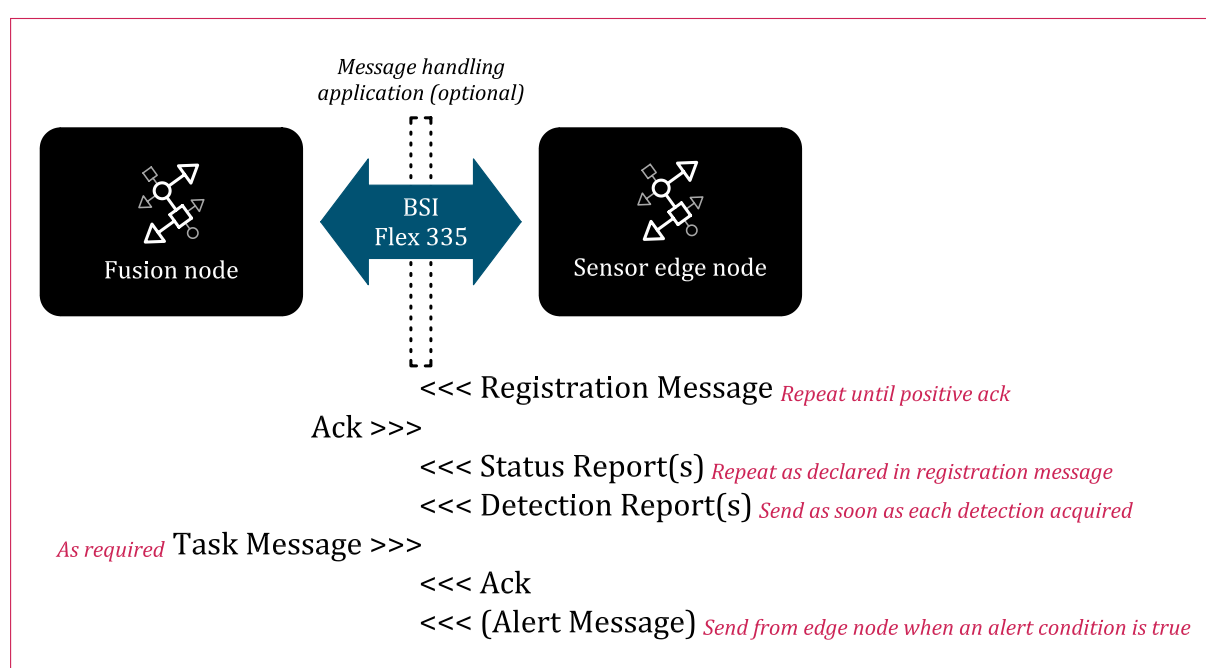
The message exchange behaviour is illustrated in **Figure 2**, which shows the following sequential activities:

- 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 Transverse 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 1122¹⁾) 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 3²⁾ 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.*

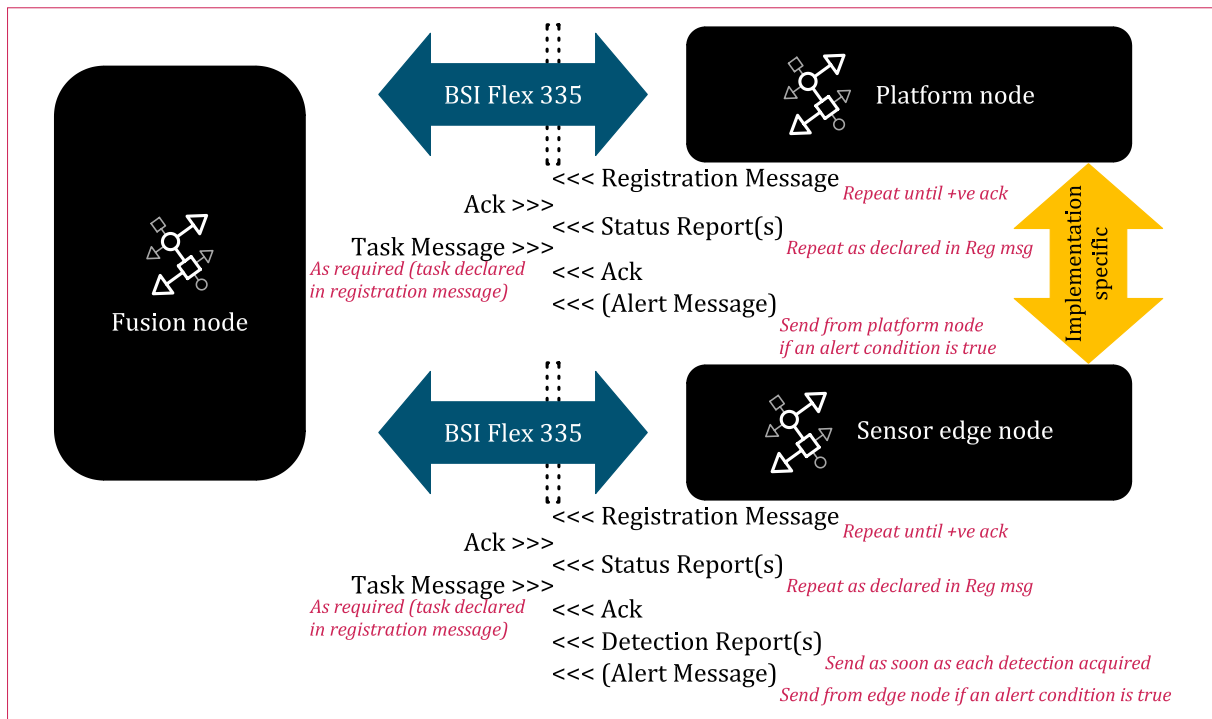
Platform nodes and dependent nodes shall declare the commands they are capable of accepting in their registration message.

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.

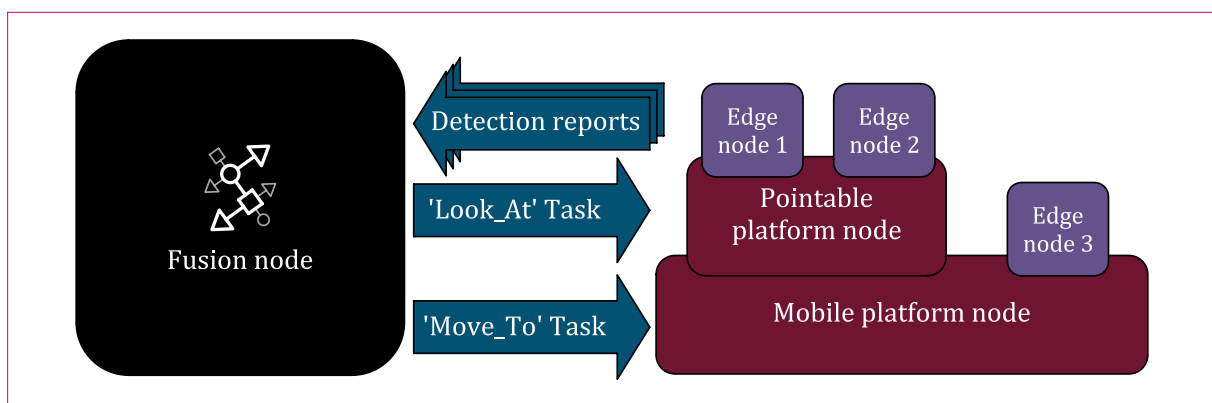
Figure 3 – Message sequence for a platform architecture



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).*

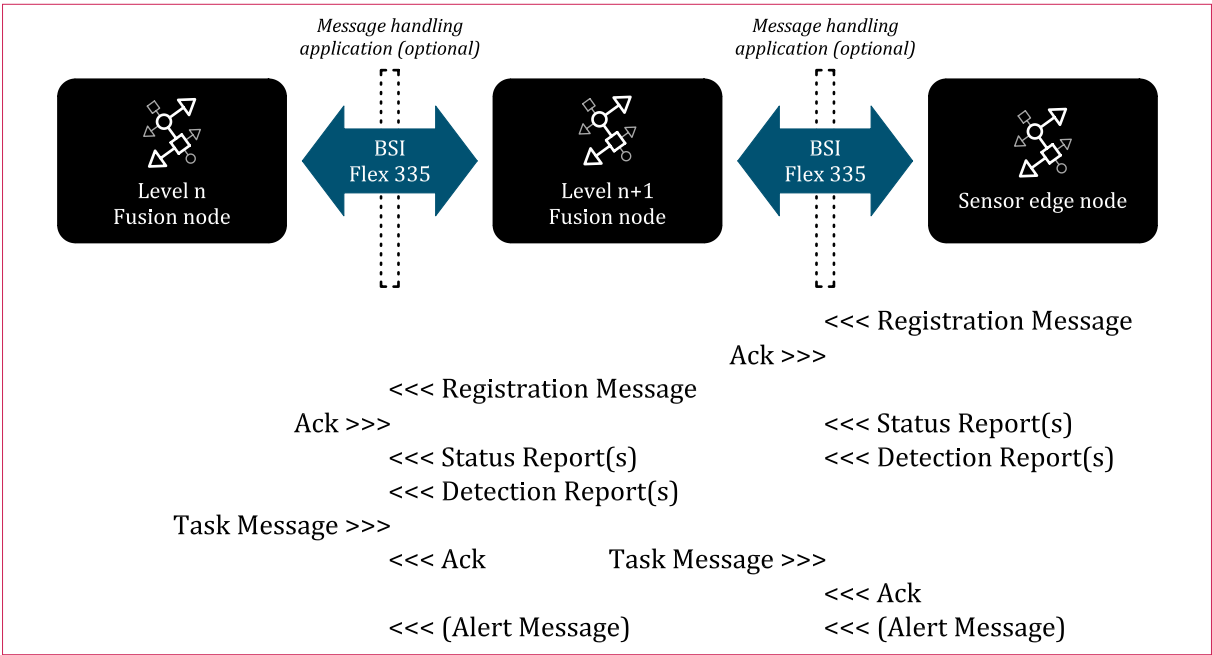
NOTE 2 *Hierarchical functionality can exist in static fusion nodes or in combination with platform nodes (see 4.10).*

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

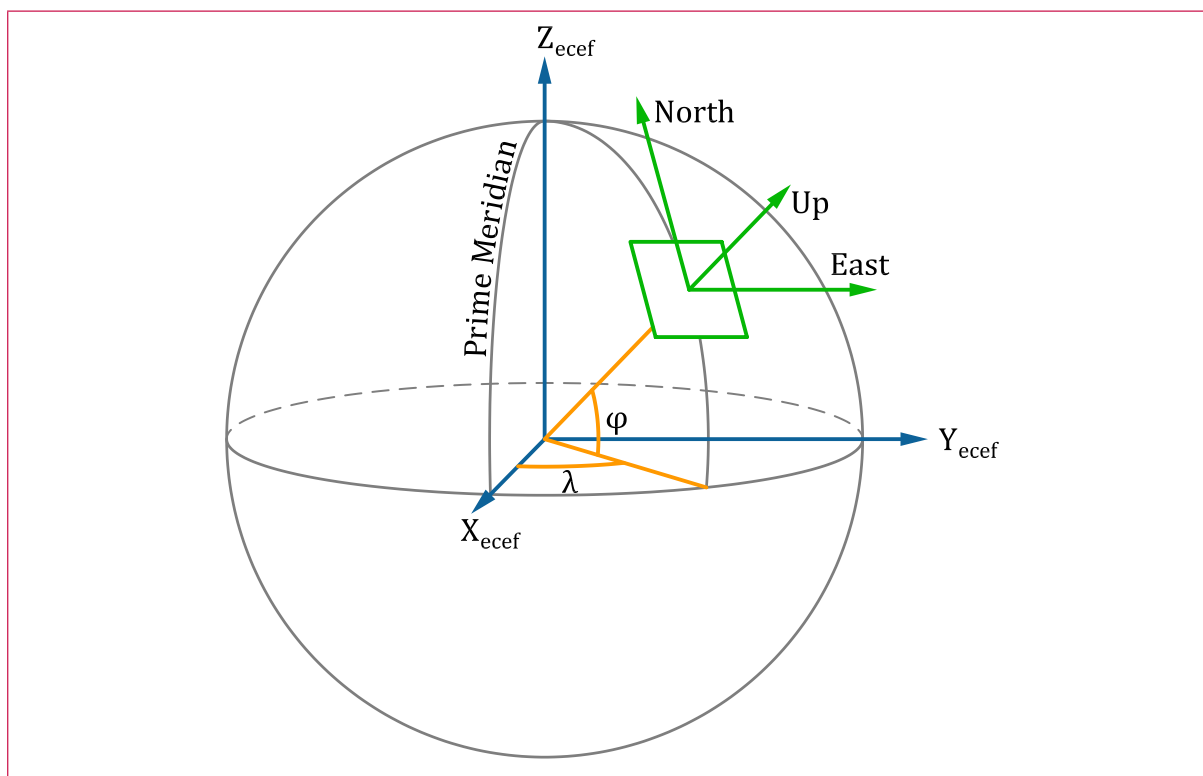
NOTE The datum for a bearing relative to a platform (Table 10) is currently not defined. This will be defined in the next edition of the standard.

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