OGC® DOCUMENT: 21-056R10

External identifier of this OGC® document: http://www.opengis.net/doc/standard/21-056r10/1.0

Open Geospatial Consortium

OGC GEOPOSE 1.0 DATA EXCHANGE DRAFT STANDARD

STANDARD
Implementation

DRAFT

Version: 1.0.0
Submission Date: 2022-02-08
Approval Date:
Publication Date:
Editor: Carl Stephen Smyth

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Suggested additions, changes and comments on this document are welcome and encouraged. Such suggestions may be submitted using the online change request form on OGC web site: http://portal.opengeospatial.org/public_ogc/change_request.php

Copyright notice

Note

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I ABSTRACT

GeoPose 1.0 is an OGC Implementation Standard for exchanging the location and orientation of real or virtual geometric objects (“Poses”) within reference frames anchored to the earth’s surface (“Geo”) or within other astronomical coordinate systems.

The standard specifies two Basic forms with no configuration options for common use cases, an Advanced form with more flexibility for more complex applications, and five composite GeoPose structures that support time series plus chain and graph structures.

These eight Standardization Targets are independent. There are no dependencies between Targets and each may be implemented as needed to support a specific use case.

The Standardization Targets share an implementation-neutral Logical Model which establishes the structure and relationships between GeoPose components and also between GeoPose data objects themselves in composite structures. Not all of the classes and properties of the Logical Model are expressed in individual Standardization Targets nor in the specific concrete data objects defined by this standard. Those elements that are expressed are denoted as implementation-neutral Structural Data Units (SDUs). SDUs are aliases for elements of the Logical Model, isolated to facilitate specification of their use in encoded GeoPose data objects for a specific Standardization Target.

For each Standardization Target, each implementation technology and corresponding encoding format defines the encoding or serialization specified in a manner appropriate to that technology.

GeoPose 1.0 specifies a single encoding in JSON format (IETF RFC 8259). Each Standardization Target has a JSON Schema (Internet-Draft draft-handrews-json-schema-02) encoding specification. The key standardization requirements specify that concrete JSON-encoded GeoPose data objects must conform to the corresponding JSON Schema definition. The individual elements identified in the encoding specification are composed of SDUs, tying the specifications back to the Logical Model.

The GeoPose 1.0 Standard makes no assumptions about the interpretation of external specifications, for example, of reference frames. Nor does it assume or constrain services or interfaces providing conversion between GeoPoses of difference types or relying on different external reference frame definitions.

II KEYWORDS

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

ogc, ogcdoc, OGC document, pose, geopose, augmented reality, ar, mixed reality, xr, reference frame

III PREFACE

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Norkart AS
- Open Site Plan
- Open AR Cloud Association
- Ordnance Survey
- School of Management and Engineering Vaud, HES-SO University of Applied Sciences and Arts Western Switzerland

VI SUBMITTERS

All questions regarding this submission should be directed to the editor or the submitters:

NAME	AFFILIATION
Nicolas Blanc	School of Management and Engineering Vaud, HES-SO University of Applied Sciences and Arts Western Switzerland
Kyoung-Sook Kim	National Institute of Advanced Industrial Science and Technology
Jeremy Morley	Ordnance Survey
Christine Perey	Open AR Cloud Association
Mahmoud Sakr	Université Libre de Bruxelles
Scott Simmons	Open Geospatial Consortium
Carl Stephen Smyth	OpenSitePlan, USA
Jan-Erik Vinje	Open AR Cloud Association

VII PARTICIPANTS IN DEVELOPMENT

The following individuals contributed to the GeoPose 1.0 development:

Table 1 — Participants in Development

NAME	INSTITUTION
Nazih Fino	Global Nomad
Josh Lieberman	Open Geospatial Consortium
Mikel Salazar	Augmented Interaction
Maxime Schoemans	Université Libre de Bruxelles
Marco Tillmann	Blackshark.ai

VIII ACKNOWLEDGEMENTS

The GeoPose Standards Working Group (SWG) wishes to thank the 3D Information Management (3DIM) Working Group of the OGC as well as Augmented City, Immersal, Trevor F. Smith, Transmutable, Khronos Group, SEDRIS, Roger Lott and Chris Little for their support and insight.

1 SCOPE

1 SCOPE

The OGC GeoPose 1.0 Standard defines requirements (rules) for the interoperable exchange of the location and orientation of real or virtual geometric objects (poses) within reference frames anchored to the earth's surface (Geo) or within other astronomical coordinate systems.

The Standard specifies:

- A basic form with no configuration options for common use cases,
- An advanced form with more flexibility for more complex applications, and
- Composite GeoPose structures to support time series chain, and graph structures.

The GeoPose Standard is based on an implementation-neutral Logical Model (LM). This LM is a formalization of a Conceptual Model (CM). The CM consists of a linked set of terms and definitions, defining a domain of discourse for the various geometric, geographic, and physical concepts related to GeoPoses. The LM formalizes the relationships between the implementable parts and aspects of the CM. The LM further establishes the structure and relationships between GeoPose components and also between GeoPoses data objects themselves in composite structures.

Note that the concrete GeoPose data objects defined by this standard correspond to only certain classes and properties of the LM. These classes and properties are identified as implementation-neutral Structural Data Units (SDUs). SDUs are aliases for the implementable elements of the LM. SDUs are grouped to define the implementation-neutral form of the GeoPose Standardization Targets: the specific implementation that the Standard addresses. For each Standardization Target, each implementation technology will have the definition of the encoding or serialization specified in a manner appropriate to that technology.

The GeoPose 1.0 Standard defines only one of many possible encoding methods for implementation of any or all of the eight Standardization Targets: JavaScript Object Notation (JSON). Each Standardization Target has a Internet-Draft draft-handrews-json-schema-02 definition. Most of the GeoPose standardization requirements are that concrete JSON GeoPose data objects shall conform to the corresponding JSON-Schema definition. The individual elements identified in the encoding specifications are SDUs that refer to one or more classes or attributes of the LM.

The GeoPose 1.0 Standard excludes assumptions about the interpretation of external specifications such as reference frames. Further, the Standard does not assume or constrain services or interfaces providing conversion between GeoPoses of difference types or relying on different external reference frame definitions.

2

CONFORMANCE

2 CONFORMANCE

Conformance with this standard shall be checked using all the relevant tests specified in Annex A of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies (https://portal.ogc.org/files/?artifact_id=55234) and Procedures and the OGC Compliance Testing web site (https://www.ogc.org/compliance). GeoPose 1.0 JSON encodings are specified via Internet-Draft draft-handrews-json-schema-02 and most of the requirements are that conforming encoded data objects shall validate against the corresponding schema. The GeoPose schemas are available from the OGC's schema repository at https://schemas.opengis.net/geopose

All requirements classes and conformance classes described in this document are owned by the standard(s) identified.

2.1. Modularity

This standard describes eight standardization targets. These targets are independent and a conforming implementation may implement one or more of the targets.

2.2. Conformance classes

This OGC® standard identifies eight conformance classes. One conformance class is defined for each corresponding set of Structural Data Units (SDUs) where each SDU is linked to the Logical Model as an alias for a class or attribute. Additionally, each of the eight standardization targets is represented by a conformance class as defined by a corresponding requirements class.

The tests in Annex A are organized by requirements class. An implementation of a conformance class must pass all tests specified in Annex A for the corresponding requirements class.

No conformance class has a dependency on another conformance class.

The Logical Model is the root normative part of this standard.

2.3. Standardization targets

There are eight independent standardization targets. Each addresses the specific requirements of one or more individual use cases. The Basic and Advanced Targets share in the use of an EPSG 4979/3D WGS-84 *outer frame* (Clause 4.2.7) but differ in the level of options and

flexibility in specification of the *inner frame* (Clause 4.2.8). The Composite Targets offer approaches to packaging sequenced or linked Frame Transforms.

The eight targets are denoted by bold terms in the following categories:

1. Basic — Satisfy most use cases — EPSG 4979 Outer Frame
    a) Local Tangent Plane East, North, Up (LTP-ENU) *inner frame* (Clause 4.2.8) oriented by Yaw, Pitch, and Roll (YPR) rotations about z, y, x axes: **Basic-YPR** Target
    b) LTP-ENU *inner frame* (Clause 4.2.8) oriented by unit quaternion: **Basic-Quaternion** Target
2. Configurable *inner frame* (Clause 4.2.8) oriented by unit quaternion — Flexible enough for complex use cases: **Advanced** Target
3. Composite — Efficient structures for linked and sequential GeoPoses
    a) Linked linear sequence of poses: **Chain** Target
    b) General linked poses: **Graph** Target
    c) Sequence
        i) Series
            A) Time series with constant time spacing: **Regular** Time series Target
            B) Time series with per-GeoPose time: **Irregular** Time series Target
        ii) Open-ended sequence of time-stamped GeoPoses: **Stream** Target

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

T. Bray (ed.): IETF RFC 8259, *The JavaScript Object Notation (JSON) Data Interchange Format*. (2017). https://www.rfc-editor.org/info/rfc8259.

Austin Wright, Henry Andrews, Ben Hutton, Greg Dennis: Internet-Draft draft-handrews-json-schema-02, *JSON Schema: A Media Type for Describing JSON Documents*. (2019). https://www.ietf.org/archive/id/draft-handrews-json-schema-02.txt.

Austin Wright, Henry Andrews, Ben Hutton: Internet-Draft draft-handrews-json-schema-validation-02, *JSON Schema Validation: A Vocabulary for Structural Validation of JSON*. (2019). https://www.ietf.org/archive/id/draft-handrews-json-schema-validation-02.txt.

ISO: ISO 19101-1:2014, *Geographic information — Reference model — Part 1: Fundamentals*. International Organization for Standardization, Geneva (2014). https://www.iso.org/standard/59164.html.

ISO: ISO 19103:2015, *Geographic information — Conceptual schema language*. International Organization for Standardization, Geneva (2015). https://www.iso.org/standard/56734.html.

ISO: ISO 19109:2015, *Geographic information — Rules for application schema*. International Organization for Standardization, Geneva (2015). https://www.iso.org/standard/59193.html.

ISO: ISO 19111:2019, *Geographic information — Referencing by coordinates*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/74039.html.

ISO: ISO 19115-1:2014, *Geographic information — Metadata — Part 1: Fundamentals*. International Organization for Standardization, Geneva (2014). https://www.iso.org/standard/53798.html.

ISO: ISO 19157:2013, *Geographic information — Data quality*. International Organization for Standardization, Geneva (2013). https://www.iso.org/standard/32575.html.

ISO: ISO/TS 19139:2007, *Geographic information — Metadata — XML schema implementation*. International Organization for Standardization, Geneva (2007). https://www.iso.org/standard/32557.html.

ISO: ISO 8601-1:2019, *Date and time — Representations for information interchange — Part 1: Basic rules*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/70907.html.

ISO: ISO 8601-2:2019, *Date and time — Representations for information interchange — Part 2: Extensions*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/70908.html.

Joan Masó and Lucy Bastin: OGC 15-097r1, *OGC® Geospatial User Feedback Standard: Conceptual Model*. Open Geospatial Consortium (2016). https://docs.ogc.org/is/15-097r1/15-097r1.html.

Gerhard Gröger, Thomas H. Kolbe, Claus Nagel, Karl-Heinz Häfele: OGC 12-019, *OGC City Geography Markup Language (CityGML) Encoding Standard*. Open Geospatial Consortium (2012). https://portal.ogc.org/files/?artifact id=47842.

Jiyeong Lee, Ki-Joune Li, Sisi Zlatanova, Thomas H. Kolbe, Claus Nagel, Thomas Becker: OGC 14-005r3, *OGC® IndoorGML*. Open Geospatial Consortium (2014). https://docs.ogc.org/is/14-005r3/14-005r3.html.

N. Freed, N. Borenstein: IETF RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. (1996). https://www.rfc-editor.org/info/rfc2045.

T. Berners-Lee, R. Fielding, L. Masinter: IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*. (2005). https://www.rfc-editor.org/info/rfc3986.

INSPIRE: D2.8.III.2 Data Specification on Buildings — Technical Guidelines. European Commission Joint Research Centre.

ISO/IEC: ISO/IEC 19505-2:2012, *Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2012). https://www.iso.org/standard/52854.html.

ISO/IEC: ISO/IEC 19507:2012, *Information technology — Object Management Group Object Constraint Language (OCL)*. International Organization for Standardization, International Electrotechnical Commission, Geneva (2012). https://www.iso.org/standard/57306.html.

Khronos Group Inc.: COLLADA — Digital Asset Schema Release 1.5.0

Cliff Kottman and Carl Reed: OGC 08-126, *Topic 5 — Features*. Open Geospatial Consortium (2009). https://portal.ogc.org/files/?artifact id=29536.

Cliff Kottman: OGC 99-108r2, *Topic 8 — Relationships Between Features*. Open Geospatial Consortium (1999). https://portal.ogc.org/files/?artifact id=894.

Thomas H. Kolbe, Tatjana Kutzner, Carl Stephen Smyth, Claus Nagel, Carsten Roensdorf, Charles Heazel: OGC 20-010, *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard*. Open Geospatial Consortium (2021). https://docs.ogc.org/is/20-010/20-010.html.

Clemens Portele: OGC 07-036, *OpenGIS Geography Markup Language (GML) Encoding Standard*. Open Geospatial Consortium (2007). https://portal.ogc.org/files/?artifact id=20509.

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the 'ModSpec'. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

4.1. Basic concepts

4.1.1. **conceptual model**

CM ADMITTED

model that defines concepts of a universe of discourse

Note 1 to entry: A conceptual model is explicitly chosen to be informed by, but be independent of design or implementation concerns.

[**SOURCE:** ISO 19101-1:2014, Clause 4.1.5]

4.1.2. **logical model**

logical information model ADMITTED

LM ADMITTED

information model that specifies the structures and relationships between data elements but is independent of any particular technology or implementation environment

[**SOURCE:** ISO 13972:2022, Clause 3.1.8]

4.1.3. **structural data unit**

SDU ADMITTED

element of the *logical model* (Clause 4.1.2) expressed in concrete data objects encoded using specific encoding or serialization technologies

4.2. Spatial concepts

4.2.1. **position**

OGC direct position ADMITTED

set of coordinates of a point in a 3D Euclidean space and associated *reference frame* (Clause 4.2.5)

4.2.2. **orientation**

rotational relationship between two *reference frames* (Clause 4.2.5)

4.2.3. **pose**

representation of a *frame transform* (Clause 4.2.6) mapping the space of an *outer frame* (Clause 4.2.7) to the space of an *inner frame* (Clause 4.2.8)

Note 1 to entry: A *pose* (Clause 4.2.3) may be associated with additional non-geometrical properties such as time of observation or validity. Poses in computer graphics often have an *outer frame* (Clause 4.2.7) defined by a parent node in a scene graph and an *inner frame* (Clause 4.2.8) define by a *position* (Annex D.26) and an orientation.

4.2.4. GeoPose

pose (Clause 4.2.3) whose associated *outer frame* (Clause 4.2.7) or a *pose chain* (Clause 4.2.17) whose associated *outermost frame* (Clause 4.2.9) is a *topocentric frame* (Clause 4.2.12) defined by an *extrinsic frame specification* (Clause 4.2.14) related to the *ephemeris object* (Clause 4.2.11) planet Earth.

4.2.5. reference frame

system of location and measurement often defined by a *frame specification* (Clause 4.2.13) usually including a coordinate system to be used within a corresponding space

4.2.6. frame transform

pair of *reference frames* (Clause 4.2.5) and a bi-continuous coordinate transformation relating points in the corresponding spaces

Note 1 to entry: The two *reference frames* (Clause 4.2.5) are called *outer frame* (Clause 4.2.7) (domain) and *inner frame* (Clause 4.2.8) (range). Only an *outer frame* (Clause 4.2.7) may have an *extrinsic frame specification* (Clause 4.2.14).

Note 2 to entry: A *frame transform* (Clause 4.2.6) functions as a directed edge in a *frame graph* (Clause 4.2.16) representation of the transformational relationship between *reference frames* (Clause 4.2.5).

4.2.7. outer frame

outer reference frame ADMITTED

first of two *reference frames* (Clause 4.2.5) associated with a *frame transform* (Clause 4.2.6)

Note 1 to entry: In the NASA SPICE system, the *outer frame* (Clause 4.2.7) is referred to as the **from** *frame* (Clause 4.2.5). In the ROS SDF documentation, the *outer frame* (Clause 4.2.7) is referred to as the **Parent** *frame* (Clause 4.2.5). In ISO 19162, the *outer frame* (Clause 4.2.7) is referred to as the **base** *frame* (Clause 4.2.5).

4.2.8. **inner frame**

inner reference frame ADMITTED

second of two *reference frames* (Clause 4.2.5) associated with a *frame transform* (Clause 4.2.6)

Note 1 to entry: An *inner frame* (Clause 4.2.8) may not be a *topocentric frame* (Clause 4.2.12).

Note 2 to entry: In the NASA SPICE system, the *inner frame* (Clause 4.2.8) is referred to as the to *frame* (Clause 4.2.5). In the ROS SDF documentation, the *inner frame* (Clause 4.2.8) is referred to as the child *frame* (Clause 4.2.5). In ISO 19162, the *inner frame* (Clause 4.2.8) is referred to as the derived *frame* (Clause 4.2.5).

4.2.9. **outermost frame**

outer frame (Clause 4.2.7) of the first *frame transform* (Clause 4.2.6) in a *pose chain* (Clause 4.2.17)

4.2.10. **innermost frame**

inner frame (Clause 4.2.8) of the last *frame transform* (Clause 4.2.6) in a *pose chain* (Clause 4.2.17)

4.2.11. **ephemeris object**

physical object or manifestation of a physical object that can be characterized by an externally-defined (possibly time-dependent) location and orientation in a 3-dimensional space

4.2.12. **topocentric frame**

topocentric reference frame ADMITTED

frame (Clause 4.2.5) that has an *extrinsic frame specification* (Clause 4.2.14) associated with a location on or near the surface of a natural body, such as planet Earth

Note 1 to entry: This definition is sourced from the NASA SPICE system.

Note 2 to entry: In connection with a GeoPose, one way that a *topocentric frame* (Clause 4.2.12) may be realized is by a *local tangent plane east-north-up frame (LTP-ENU)* (Annex D.24) attached to the surface of a body, to a gravitational equipotential surface (*geoid* (Annex D.21) in the case of planet Earth), or to a mathematical surface such as an *ellipsoid* (Annex D.16) approximating a *geoid* (Annex D.21).

4.2.13. **frame specification**

data that completely and uniquely defines a *reference frame* (Clause 4.2.5)

Note 1 to entry: In the context of *poses* (Clause 4.2.3), there are *extrinsic frame specification* (Clause 4.2.14) defined by an external data source, and *derived frame specification* (Clause 4.2.15) defined by a transformation from another *reference frame* (Clause 4.2.5).

4.2.14. **extrinsic frame specification**

extrinsic specification ADMITTED

relates a *reference frame* (Clause 4.2.5) to an *ephemeris object* (Clause 4.2.11) or other external reference, which may be based on joint properties of a group of objects

Example The center of mass of the Earth-Moon system.

4.2.15. **derived frame specification**

derived specification ADMITTED

relates a *reference frame* (Clause 4.2.5) to another *reference frame* (Clause 4.2.5) by a *frame transform* (Clause 4.2.6) or its inverse

4.2.16. **frame graph**

reference frame graph ADMITTED

directed acyclic graph representation of the transformational relationships between *reference frames* (Clause 4.2.5)

Note 1 to entry: In the frame graph, *reference frames* (Clause 4.2.5) are the nodes or vertices of the graph. *Frame transforms* (Clause 4.2.6) are the edges of the graph, directed from the *outer frame* (Clause 4.2.7) to the *inner frame* (Clause 4.2.8). Note that there may be zero, one, or many paths between two distinct vertices, i.e. *frames* (Clause 4.2.5). Multiple paths correspond to real-world situations with, for example, redundant line-of-sight links in point-to-point radio networks used in communication systems.

4.2.17. **pose chain**

directed path in a *frame graph* (Clause 4.2.16) connecting an *outermost frame* (Clause 4.2.9) to an *innermost frame* (Clause 4.2.10)

Note 1 to entry: The sequence of *frame transforms* (Clause 4.2.6) in a *pose chain* (Clause 4.2.17) may be combined in a single composite transformation.

Note 2 to entry: There may exist multiple *pose chains* (Clause 4.2.17) linking the same *outermost frame* (Clause 4.2.9) and *innermost frame* (Clause 4.2.10) and the corresponding composite transformations may not agree. This is intentional, representing real-world configurations and capabilities of sensors and communication links.

4.3. Sequence and stream concepts

4.3.1. **sequence**

GeoPose sequence ADMITTED

set of *poses* (Clause 4.2.3) ordered by *valid time* (Clause 4.5.1) and pertaining to the same underlying physical object or construct

Note 1 to entry: A *pose* (Clause 4.2.3) in a sequence is called a "member pose".

Note 2 to entry: In a sequence, each successive member *pose* (Clause 4.2.3) must have a *valid time* (Clause 4.5.1) after its predecessor.

4.3.2. inter-pose duration

time *duration* (Clause 4.4.5) between consecutive *poses* (Clause 4.2.3) in a *sequence* (Clause 4.3.1)

4.3.3. closed sequence

closed pose sequence ADMITTED

sequence (Clause 4.3.1) of fixed length with specific meta-data that fully characterize the sequence and its member *poses* (Clause 4.2.3)

4.3.4. regular sequence

regular GeoPose sequence ADMITTED

closed sequence (Clause 4.3.3) with a constant *inter-pose duration* (Clause 4.3.2)

4.3.5. irregular sequence

irregular GeoPose sequence ADMITTED

closed sequence (Clause 4.3.3) with a variable *inter-pose duration* (Clause 4.3.2)

Note 1 to entry: Each *pose* (Clause 4.2.3) in an *irregular sequence* (Clause 4.3.5) has an associated *valid time* (Clause 4.5.1).

4.3.6. GeoPose stream

irregular sequence (Clause 4.3.5) of unbounded length

4.3.7. header

sequence header ADMITTED

metadata essential for interpretation of the following members of a *sequence* (Clause 4.3.1)

4.3.8. transition model

metadata that indicates whether or how it may be possible to estimate *poses* (Clause 4.2.3) in the interval between consecutive *poses* (Clause 4.2.3) in a *sequence* (Clause 4.3.1)

4.3.9. trailer

sequence trailer ADMITTED

metadata essential for validation of the preceding members of a *sequence* (Clause 4.3.1).

4.4. Temporal concepts

4.4.1. temporal frame

specification for the interpretation of points on a *time line* (Clause 4.4.2) as *instants* (Clause 4.4.3) in relation to a specified *epoch* (Clause 4.4.6)

4.4.2. time line

time axis ADMITTED

one-dimensional Euclidean space whose points represent an ordered sequence of *instants* (Clause 4.4.3) directed from the past to the future

4.4.3. instant

specific point on a *time line* (Clause 4.4.2)

4.4.4. interval

timespan between two *instants* (Clause 4.4.3) on a *time line* (Clause 4.4.2), interpreted in context of the associated *temporal frame* (Clause 4.4.1)

4.4.5. duration

one-dimensional signed distance between the bounding *instants* (Clause 4.4.3) of an *interval* (Clause 4.4.4)

Note 1 to entry: The magnitude of a length value depends on the *temporal frame* (Clause 4.4.1).

Note 2 to entry: A duration is semi-open: it includes the earlier *instant* (Clause 4.4.3) but not the later *instant* (Clause 4.4.3).

4.4.6. epoch

specified *instant* (Clause 4.4.3) that can be used as a reference point to calculate *temporal relationships* (Clause 4.4.7) and *durations* (Clause 4.4.5) between *instants* (Clause 4.4.3).

4.4.7. **temporal relationship**

relationship between two *instants* (Clause 4.4.3)

Note 1 to entry: Temporal relationships are only valid within the context of a specific *temporal frame* (Clause 4.4.1).

Note 2 to entry: GeoPose supports three temporal relationships: “before”, “coincident”, and “after”.

4.5. Temporal database concepts

4.5.1. **valid time**

time line (Clause 4.4.2) where the time of changes in the existence or validity of real-world objects or property values are located.

Note 1 to entry: *Instants* (Clause 4.4.3) in *valid time* (Clause 4.5.1) mark the temporal location of real-world transitions in existence, property values, or their validity.

Note 2 to entry: This term may refer to *instants* (Clause 4.4.3) or to *time lines* (Clause 4.4.2).

4.5.2. **transaction time**

time line (Clause 4.4.2) where the time of changes in the presence or validity of the representations of real-world objects or their properties in an information system are located

Note 1 to entry: *Instants* (Clause 4.4.3) in *transaction time* (Clause 4.5.2) mark the temporal location of actions that create, update, or delete representations of objects or properties.

Note 2 to entry: This term may refer to *instants* (Clause 4.4.3) or to *time lines* (Clause 4.4.2).

4.5.3. **bi-temporality**

property of a data representation that denotes that it carries both *valid time* (Clause 4.5.1) and *transaction time* (Clause 4.5.2)

5

CONVENTIONS

5 CONVENTIONS

This section provides details and examples for conventions used in this document.

5.1. Identifiers

The normative provisions in this document are denoted by the URI

http://www.opengis.net/spec/GeoPose/1.0

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

5.2. Use Cases, Concepts, Logical Model, Standardization Targets, Encodings

5.3. UML Notation

The logical structure of the elements used in the GeoPose Standard is presented in this document in diagrams using the Unified Modeling Language (UML) static structure diagram (see Booch et al. [10]). The UML notations used in this standard are described in the diagram in Figure 1.

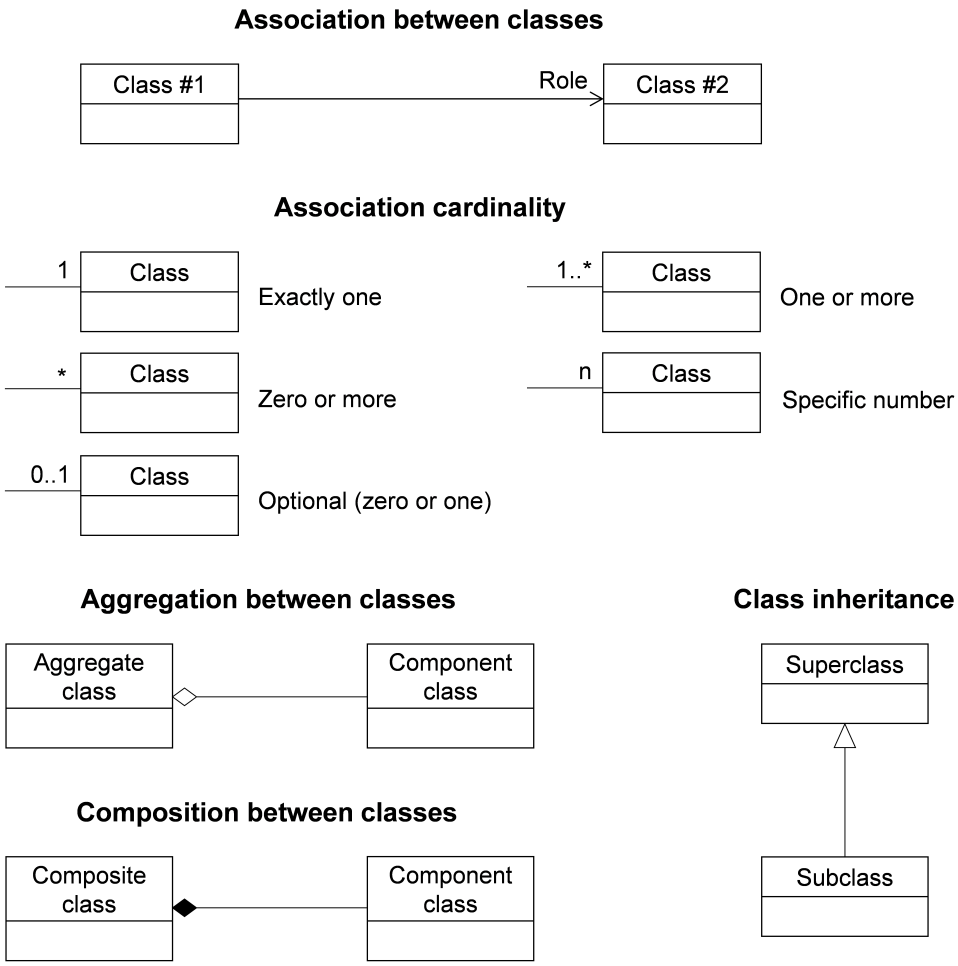

Figure 1 — UML notation (ISO 19103:2015)

All associations between model elements in GeoPose are uni-directional. Thus, associations in GeoPose are navigable in only one direction. The direction of navigation is depicted by an arrowhead. In general, the context an element takes within the association is indicated by its role. The role is displayed near the target of the association. If the graphical representation is ambiguous though, the position of the role has to be drawn to the element the association points to.

In order to enhance the readability of the GeoPose UML diagrams, classes are depicted in different colors. The following coloring scheme is applied:

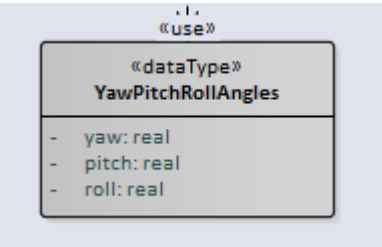

Figure 2 — Data Types

Classes painted in grey represent data types.

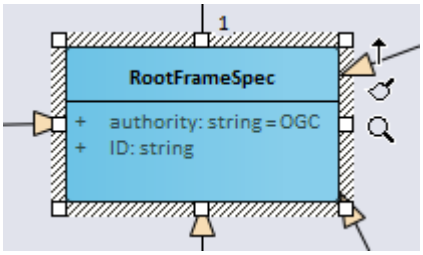

Figure 3 — Blue Color Denotes Abstract or More General Classes

Classes painted in blue are internal less-derived elements that are not themselves directly encoded as concrete data objects.

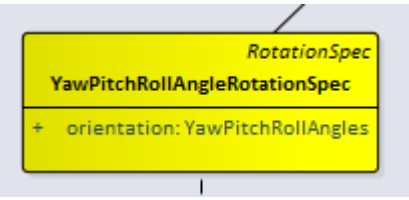

Figure 4 — Yellow Color Denotes Structural Data Units

Classes painted in yellow correspond to Structural Data Units or have properties that are represented in Structural Data Units and in encodings of those SDUs. Only data types and classes painted yellow are encoded as concrete data objects.

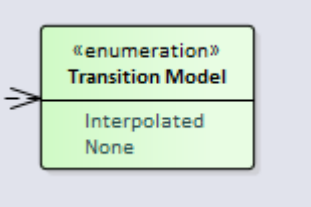

Figure 5 — Green Color Denotes Enumeration

Enumerations are painted in green.

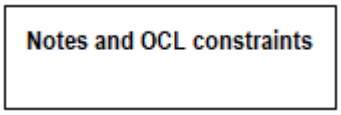

Figure 6 — White Color Denotes a Note or Constraint

The color white is used for notes and OCL constraints that are provided in the UML diagrams.

6

CONCEPTUAL MODEL

6 CONCEPTUAL MODEL

6.1. General

ISO 19101-1:2014 defines a universe of discourse to be a view of the real or hypothetical world that includes everything of interest. That standard defines a conceptual model to be a model that defines concepts of a universe of discourse.

The goal of this GeoPose Standard is to establish and document a common set of concepts that spans the targeted use cases. This does not attempt to redefine application concepts, but merely to present a common set of concepts from and to which their concepts can be understood and mapped.

The GeoPose Conceptual Model (CM) is a graph of related concepts. One technology-independent realization is given by the GeoPose Logical Model (LM).

The GeoPose CM consists of linked definitions of terms denoting concepts expressed in the GeoPose LM and Structural Data Unit (SDU) specifications for the standardization targets.

The CM describes a (non-normative) domain of discourse for terms used in defining a precise and normative LM expressed as a Unified Modelling Language (UML) class diagram.

The scope of the standardization targets is a subset of the scope of the LM. The scope of the LM is a subset of the scope of the CM. The standardization targets are mutually independent implementations of subsets of the LM.

6.2. Temporal concepts and reference frame

The only temporal frame used in this GeoPose standard is “Unix Time”: seconds since the Unix Epoch of 1 January 1970 measured by a virtual “Unix clock”, ticking once per “Unix second”, and omitting any corrections such as leap seconds.

Times before 1 January 1972 are not precisely related to another temporal frame but the value at UTC 1 January 1972 was +63,072,000. This allows precise conversion to and from modern temporal frames.

NOTE: This standard does not reference a calendar and encoded values are representations of the count of seconds, rather than a calendar-relative date and time. These times may be converted to UTC and expressed as text (e.g. with ISO 8601-1:2019 and ISO 8601-2:2019) relative to a specific calendar but this is outside the scope of this standard.

Temporal concepts defined in this document are intended to align with terms used in W3C CR-owl-time-20200326.

6.3. Document structure

The structure of the GeoPose Standard document flows from:

- use cases to the definition of a conceptual domain of discourse comprehensive enough to support those use cases,
- a realization of a portion of that conceptual domain with an implementation-neutral but specific and normative logical data model expressed in UML, and
- the normative derivation of specific structural data units that represent abstract implementation and standardization targets.

These Structural Data Units (SDUs) are abstract: they are independent of implementation or delivery technology and serialization or encoding formats. GeoPose 1.0 specifies one of many possible realizations of the structural data units in JSON.

A key aspect of the GeoPose Standard is that specific use cases are tied to the standardization targets, which prescribe the structure and content of GeoPose data objects. Corresponding implementation examples appear in other documents.

Of course, GeoPose must incorporate or align with other relevant existing standards and common practices. The goal is to fill an interoperability gap in existing standards without reinventing technology in a way that encourages interoperability.

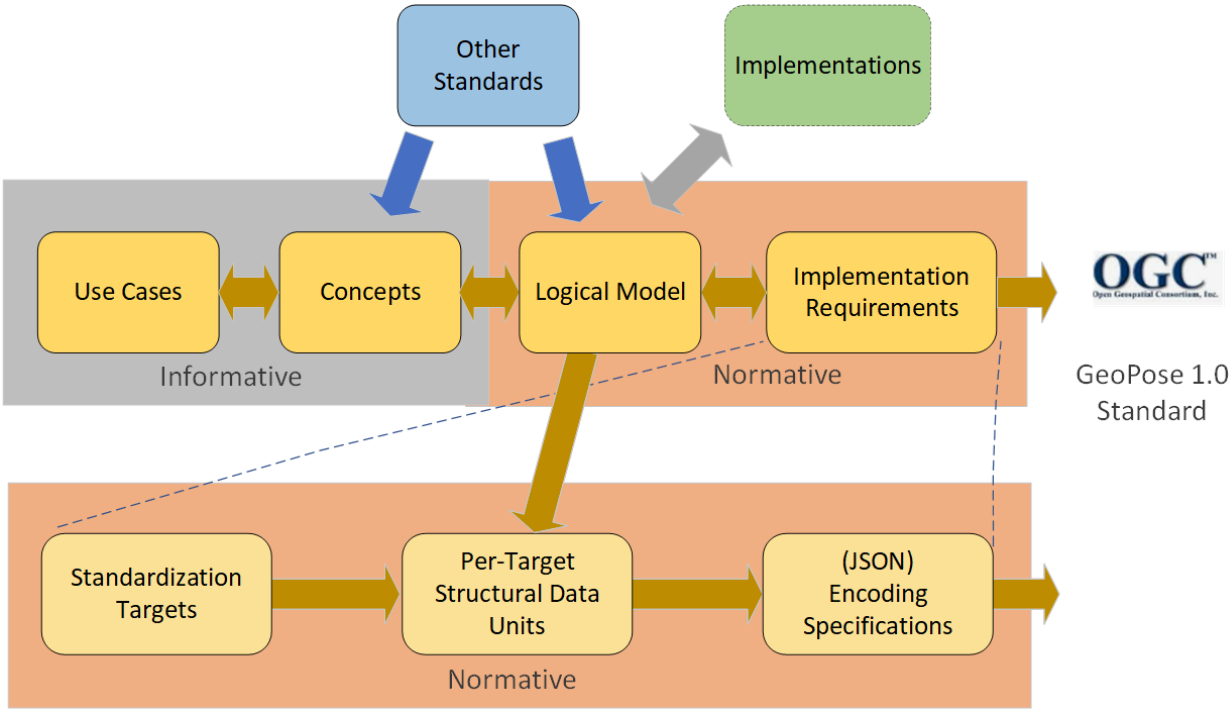

Figure 7 — Document Structure Overview

6.4. Use Case Summary

The GeoPose use cases involve interactions between information systems or between an information system and a storage medium. The essential role of a GeoPose is to convey the position and orientation of a real or virtual object. The possibility of chained transformational relationships and cross-linkages between chains affords representation of complex pose relationships and a way to bring a collection of related GeoPoses in a common geographic reference frame.

Each use case is identified by a unique ID, has a brief description, and a list of the relevant standardization targets.

6.4.1. Augmented and Mixed Reality [AR]

Augmented Reality (AR) integrates synthetic objects or synthetic representations of real objects with a physical environment. Geospatial AR experiences can use GeoPose to position synthetic objects or their representations in the physical environment. The geospatial connection provides a common reference frame to support integration in AR.

Table 2 — GeoPose use cases for Augmented and Mixed Reality

ID	DESCRIPTION	STANDARDIZATION TARGET
/geopose/1.0/use_case/ar/01	Stored representation of synthetic objects	Basic-YPR, Basic-Quaternion, Advanced
/geopose/1.0/use_case/ar/02	Positioning information to support integration of synthetic object data in a representation or visualization of the physical environment	Basic-YPR, Basic-Quaternion, Advanced
/geopose/1.0/use_case/ar/03	Report of position and orientation from a mobile device to an AR network service	Advanced (time)
/geopose/1.0/use_case/ar/04	Input to visual occlusion calculations	Basic-YPR, Basic-Quaternion
/geopose/1.0/use_case/ar/05	Input to ray-casting and line-of-sight calculations	Basic-YPR, Basic-Quaternion, Chain
/geopose/1.0/use_case/ar/06	Input to proximity calculations Basic-YPR, Basic-Quaternion,	/geopose/1.0/use_case/ar/07

6.4.2. Autonomous Vehicles [AV]

Autonomous vehicles are mobile objects that move through water, across a water surface, in the air, through the solid earth (tunnel boring machine), on the land surface, or in outer space without real-time control by an independent onboard operator. A pose captures the essential information in positioning and orienting a moving object. Sensors attached to mobile elements have their own poses and a chain of reference frame transformations enables common reference frames to be used for data fusion. The possibility of relating the vehicle to other elements of the environment via a common reference frame is essential.

Table 3 — GeoPose use cases for Autonomous Vehicles

ID	DESCRIPTION	STANDARDIZATION TARGET
/geopose/1.0/use_case/av/01	Provide accurate visual positioning and guidance based on one or more services based on a 3D representation of the real world combined with real time detection and location of real world objects	Basic-YPR, Basic-Quaternion
/geopose/1.0/use_case/av/02	Calculate parameters such as distances and routes	Basic-YPR, Basic-Quaternion, Regular Timeseries, Irregular Timeseries, Stream
/geopose/1.0/use_case/av/03	Record the trajectory of a moving vehicle.	Regular Timeseries, Irregular Timeseries, Stream

6.4.3. Built Environment [BE]

The built environment consists of objects constructed by humans and located in physical space. Buildings, roads, dams, railways, and underground utilities are all part of the built environment. The location and orientation of built objects, especially those whose view is occluded by other objects is essential information needed for human interaction with the built environment. A common reference frame tied to the earth's surface facilitates the integration of these objects when their representations are supplied by different sources.

Table 4 — GeoPose use cases for the Built Environment

ID	DESCRIPTION	STANDARDIZATION TARGET
/geopose/1.0/use_case/be/01	Specify the position and orientation of visible objects and objects that are underground or hidden within a construction.	Basic-YPR, Basic-Quaternion
/geopose/1.0/use_case/be/02	Compactly and consistently specify or share the location and pose of objects in architecture, design and construction.	Basic-YPR, Basic-Quaternion

6.4.4. Synthetic Environments [SE]

Synthetic environments contain collections of moving objects, which themselves may be composed of connected and articulated parts, in an animation or simulation environment that contains a fixed background of air, land, water, vegetation, built objects, and other non-moving elements. The assembly is animated over some time period to provide visualizations or analytical results of the evolving state of the modelled environment. Synthetic environments support training, rehearsal, and archival of activities and events. The location and orientation of the movable elements of a scene are the key data controlling animation of in a synthetic environment. Since there may be multiple possible animations consistent with observations, storage of the sequences of poses of the actors, vehicles, and other objects is a direct and compact way of representing the variable aspects of the event. Access to one or more common reference frames through a graph of frame transformations makes a coherent assembly possible.

Table 5 — GeoPose use cases for Synthetic Environments

ID	DESCRIPTION	STANDARDIZATION TARGET
/geopose/1.0/use_case/se/01	Record pose relationships of all mobile elements in an environment	Graph
/geopose/1.0/use_case/se/02	Control animation of mobile elements in an environment using stored pose time sequences	Graph, Regular Timeseries, Irregular Timeseries, Stream

6.4.5. Image Understanding [IM]

Image understanding is the segmentation of an image or sequence of images into inferred 3D objects in specific semantic categories, possibly determining or constraining their motion and/or geometry. One important application of image understanding is the recognition of moving elements in a time series of images. A pose is a compact representation of the key geometric characteristics of a moving element. In addition to moving elements sensed by an imaging device, it is often useful to know the pose of the sensor or imaging device itself. A common geographic reference frame integrates the objects into a single environment.

Table 6 — GeoPose use cases for Image Understanding

ID	DESCRIPTION	IMPLEMENTATION TARGET
/geopose/1.0/use_case/im/01	Instantaneous and time series locations and orientations of mobile objects	Basic-YPR, Basic-Quaternion, Advanced, Regular Timeseries, Irregular Timeseries, Stream
/geopose/1.0/use_case/im/02	Instantaneous and time series location and orientation of an optical imaging device using Simultaneous Location And Mapping (SLAM)	Basic-YPR, Basic-Quaternion, Advanced, Regular Timeseries, Irregular Timeseries, Stream
/geopose/1.0/use_case/im/03	Instantaneous and time series estimation of the changes in location and orientation of an object using an optical imaging device (Visual Odometry)	Basic-YPR, Basic-Quaternion, Advanced, Regular Timeseries, Irregular Timeseries, Stream
/geopose/1.0/use_case/im/04	Instantaneous and time series location and orientation of an optical imaging device used for photogrammetry	Regular Timeseries, Irregular Timeseries, Stream

7

LOGICAL MODEL

7 LOGICAL MODEL

7.1. General

The Frame Transform is the core abstraction in the GeoPose Standard. The Frame Transform is a representation of the transformation taking an Outer Frame coordinate system to an Inner Frame coordinate system. This abstraction is constrained in GeoPose v1.0 to only allow transformations involving translation and rotation. The intention is to match the usual concept of a pose as a position and orientation. The formalism that expresses a GeoPose Frame Transform is a pair of Reference Frames, Outer and Inner, each defined by a Frame Specification. The Logical Model relates these elements to represent different types of GeoPose data objects and also defines structures built of time series and linked GeoPoses.

7.2. UML Logical Model

The normative expression of the UML model is a Sparx Systems Enterprise Explorer project ("eapx") file located at:

- https://schemas.opengis.net/geopose/1.0/Model.eapx

The Logical Model consists of four top-level packages: Core, Time, Sequence, and Targets. The Targets package contains two detail packages: Basic and Composite. The Composite package is in turn subdivided into a Linked package and a Sequence package. The Basic GeoPose targets depend on only the Core package. The Advanced GeoPose target also depends on the Time Package. Composite GeoPoses depend on all four top-level packages.

The coloring of the classes indicates their role in the logical design. Note that the classes and data types defined in the Target packages are the source of structural data units (SDUs) that may be realized as concrete data objects.

7.2.1. Core

The Logical Model Core contains the essential elements specific to the GeoPose modelled as a transformation between an anchoring Outer Frame and one or more derived Inner Frames. This is described in Figure 8.

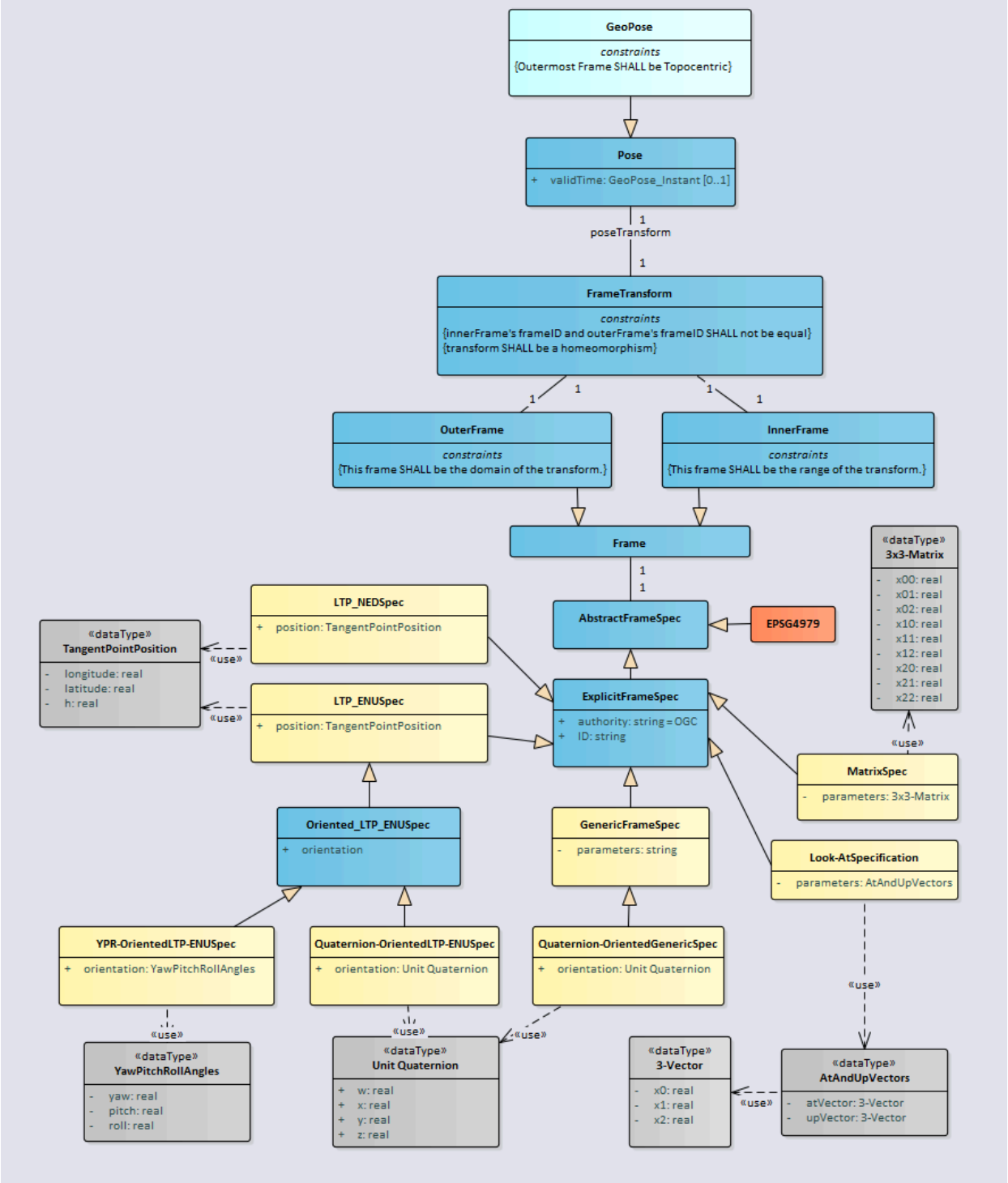

Figure 8 — Core logical model

7.2.2. Time

The time logical model is based on W3C CR-owl-time-20200326.

Only relevant classes, properties, and associations are included. GeoPose v1.0 has a very restricted idea of time position, limited to seconds of UNIX Time. This is described in Figure 9.

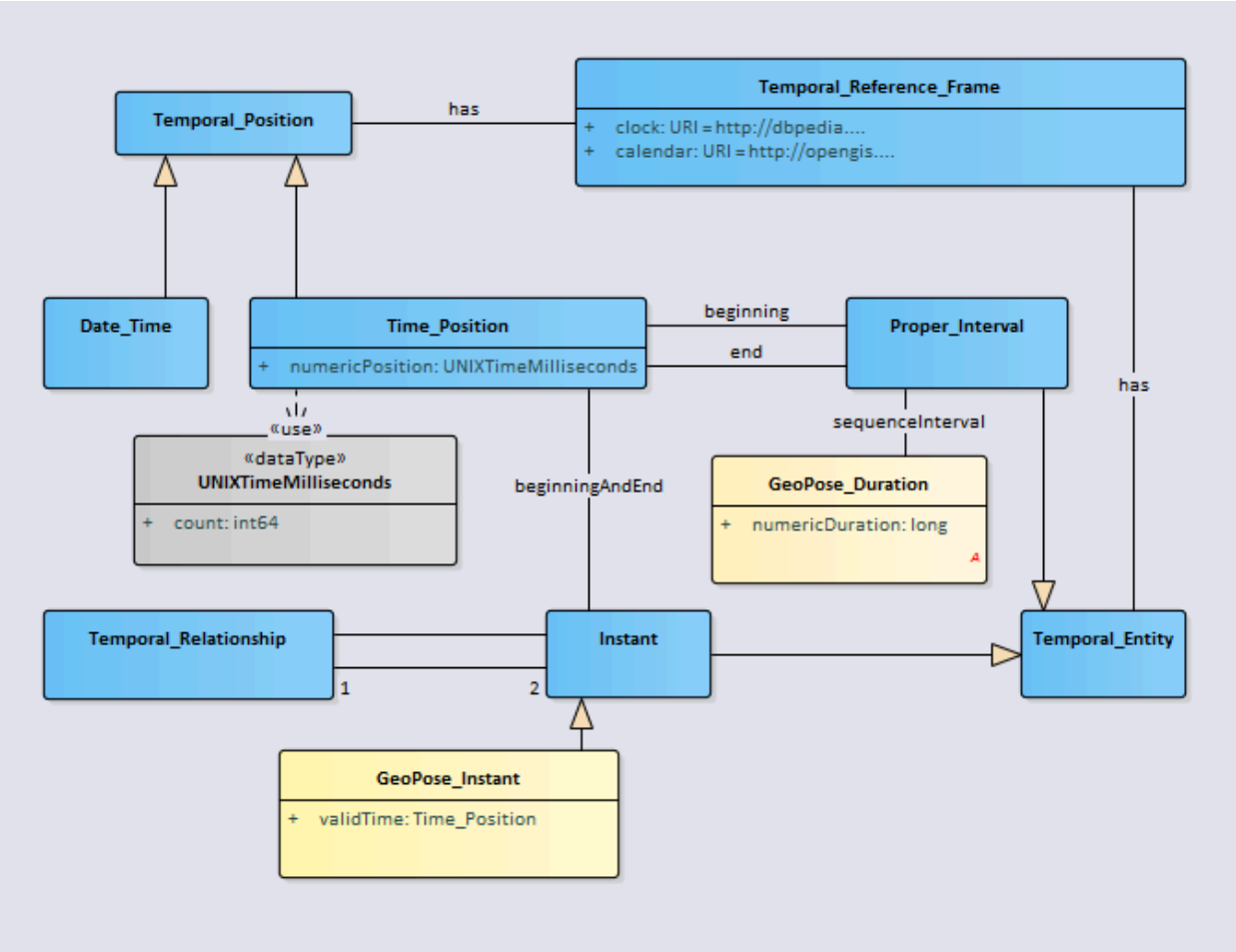

Figure 9 — Time logical model

7.2.3. Sequence

The sequence logical model defines a method for packaging of GeoPose data, where multiple GeoPoses in a sequence share the same *outer frame* (Clause 4.2.7) and there is a time-dependent changing *inner frame* (Clause 4.2.8). This is displayed in Figure 10.

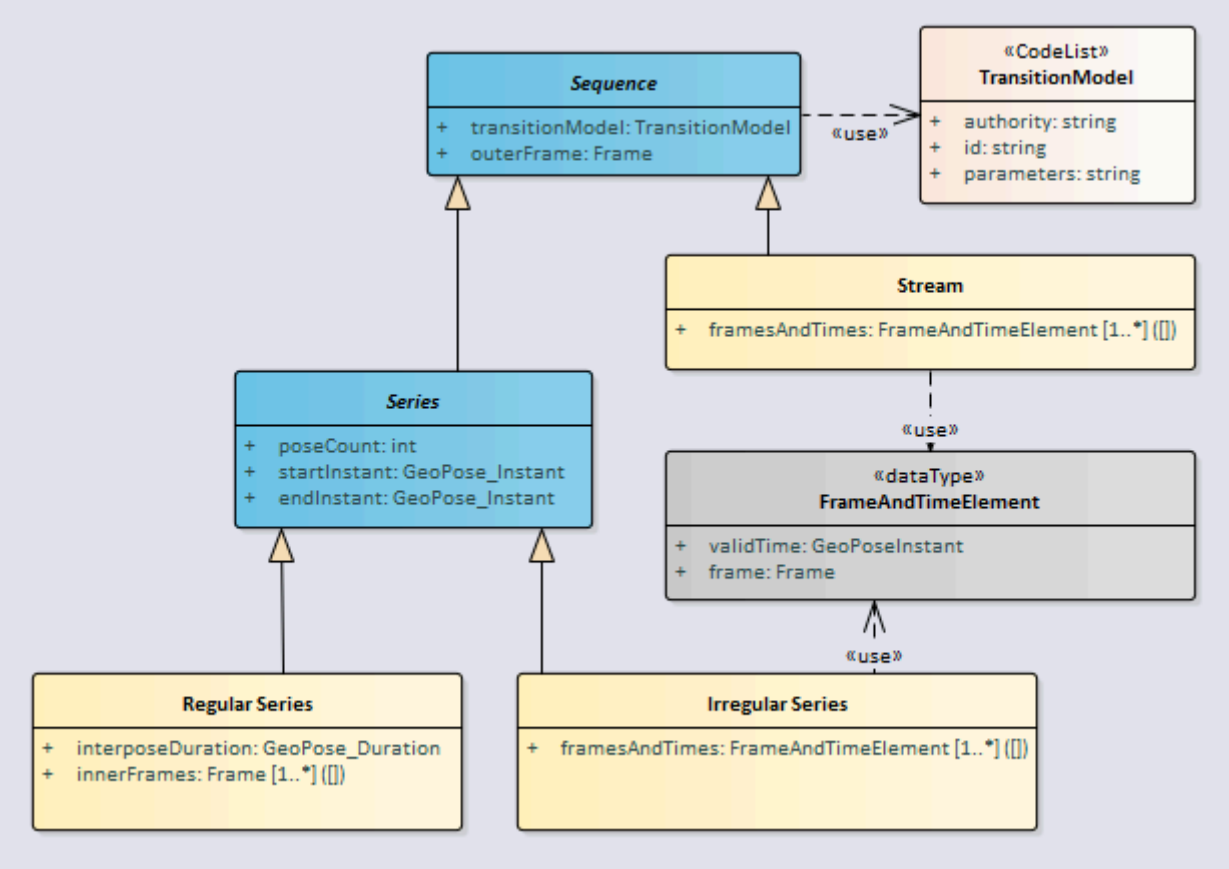

Figure 10 — Sequence logical model

7.2.4. Targets

The Logical Model's Targets package specify the design of logical data objects and data types that are directly expressed in GeoPose data objects.

The Basic-YPR, Basic-Quaternion, and the Advanced GeoPose SDUs represent single GeoPose objects.

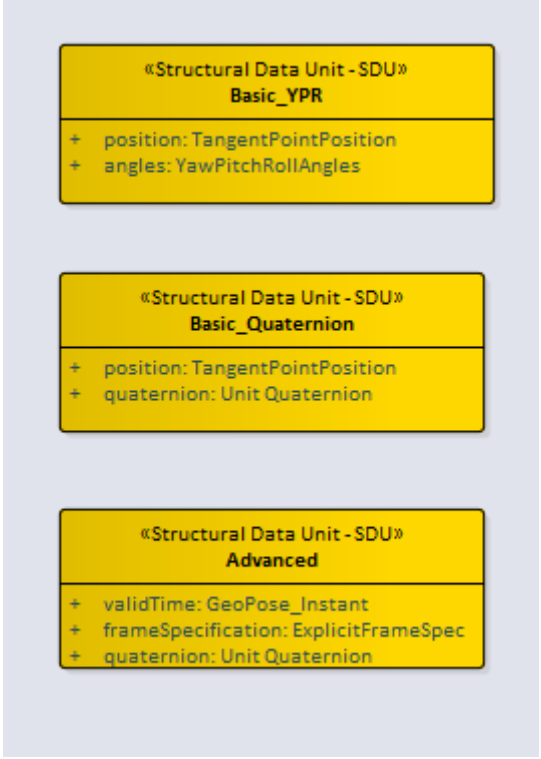

Figure 11 — Basic and Advanced Structural Data Units

The Chain and the Graph GeoPose composite structures respectively represent linear or branching frame transformation relationships.

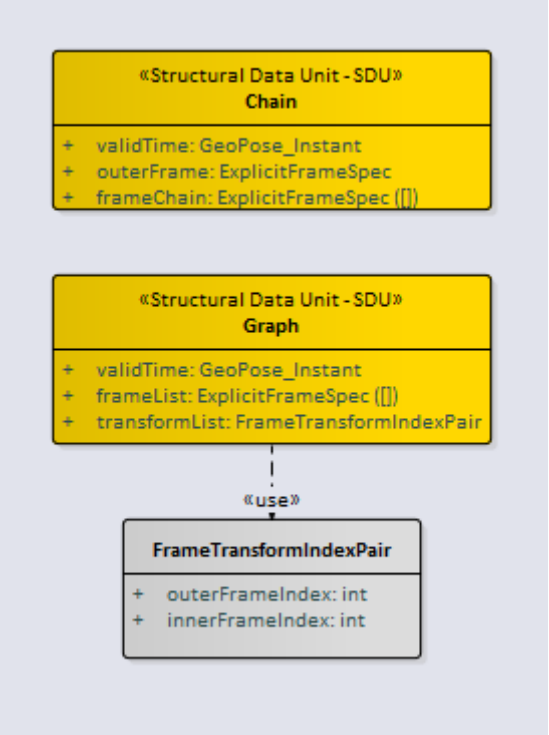

Figure 12 — Chain and Graph Structural Data Units

The Stream and each of the two Series composite structures represent time series of a single evolving GeoPose.

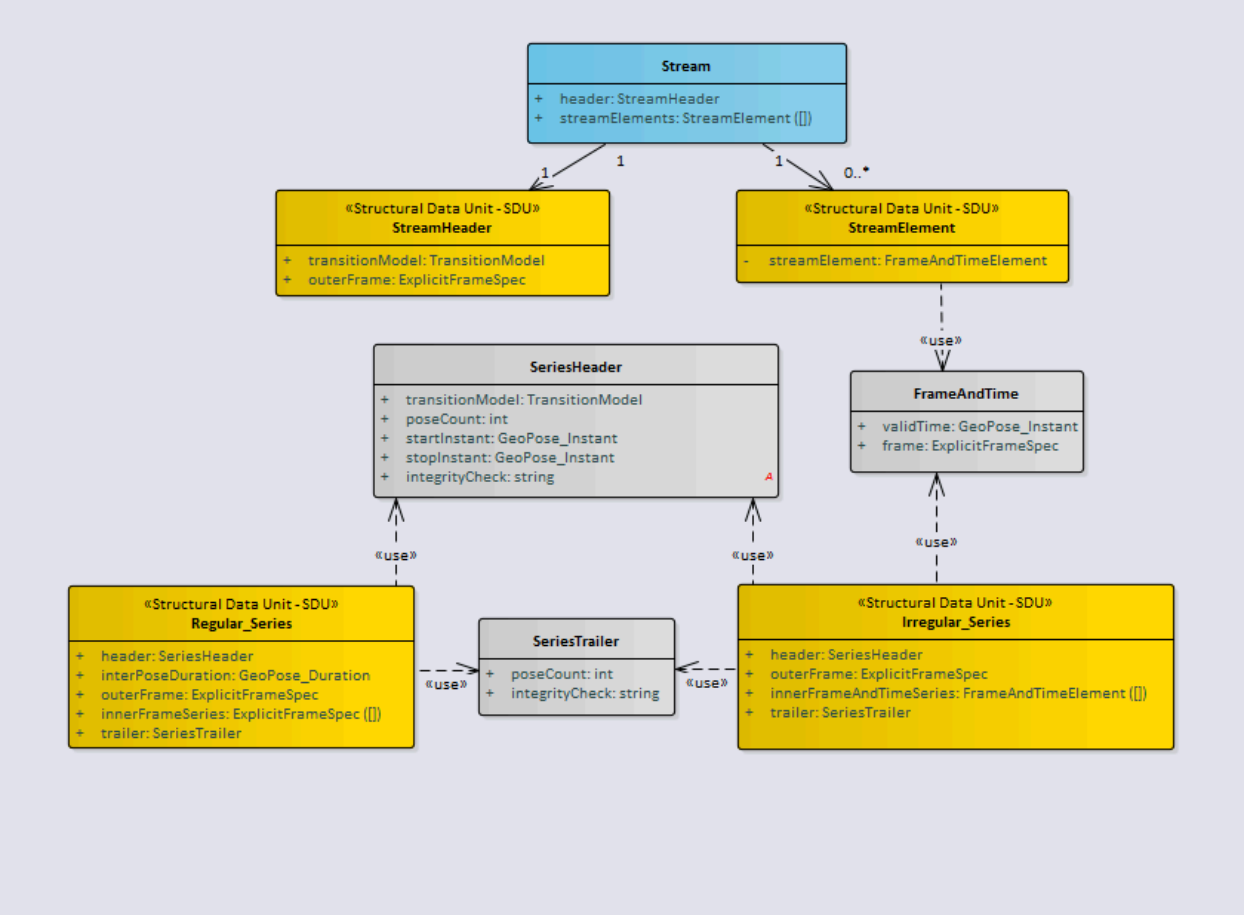

Figure 13 — Series and Stream Structural Data Units

NOTE: The *integrityCheck* attributes in the SeriesHeader and SeriesTrailer classes are defined as strings and have no prescribed method of use in GeoPose 1.0. They are placeholders to allow experimentation and possible standardization in a later version.

8

STRUCTURAL DATA UNITS AND STANDARDIZATION TARGETS

8 STRUCTURAL DATA UNITS AND STANDARDIZATION TARGETS

8.1. General

Classes, attributes, and relationships of the GeoPose domain are specified in a (normative) GeoPose UML static class model — the GeoPose Logical Model. Standardization Targets are specified by encoding-neutral elements of the Logical Model. These Structural Data Units (SDUs) are elements (classes or attributes) in the Logical Model with the "Structural Data Unit — SDU" stereotype. SDUs may have additional Requirements limiting the range, multiplicity, representation or other constraining and testable characteristics. SDUs are used individually or in combination combined to express each of the Standardization Targets.

SDUs provide Standardization Targets that are independent of serialization/encoding format. This allows multiple equivalent serializations to be defined. Each SDU that may be expressed as a concrete data object is associated with a corresponding element (class or attribute) in the logical model.

The Basic and Advanced Standardization Targets differ in the level of options and flexibility in the Frame Specifications. The Composite Targets offer approaches to packaging Frame Transforms. The Targets are the data classes that are specified by the GeoPose Standard. There are eight Standardization Targets denoted by bold terms in the following categories:

1. Basic — Satisfy most use cases
    a) Orientation by Yaw, Pitch, and Roll (YPR) rotations about z, y, x axes: **Basic-YPR** Target
    b) Orientation by unit quaternion: **Basic-Quaternion** Target
2. Configurable — Flexible enough for complex use cases including full 6DoF transformations: **Advanced** Target

3. Composite — Efficient structures for linked and sequential GeoPoses
    a) Linked linear sequence of poses linked by full 6DoF transformations: **Chain** Target
    b) General linked poses: **Graph** Target
    c) Sequence
        i) Series
            A) Time series with constant time spacing: **Regular** Time series Target
            B) Time series with per-GeoPose time: **Irregular** Time series Target
        ii) Open-ended sequence of time-stamped GeoPoses: **Stream** Target

NOTE: The definition of a reference frame by an external standard in **not** specified. GeoPose does use a three-part designation of an external frame specification using the three fields *authority*, *ID*, and *parameters*. The interpretation of the contents of these fields is outside the scope of GeoPose.

8.2. Global requirements

Global requirements apply to all SDUs and Standardization Targets.

REQUIREMENTS CLASS 1: GLOBAL SDU REQUIREMENTS	
`/req/global`	
Conformance test	Conformance class A.1: `/conf/global`
Global requirements apply to all SDUs and Standardization Targets.	
Requirement	Conformance test A.1: `/req/global/target-independence`
Requirement	Conformance test A.2: `/req/global/logical-model`
Requirement	Conformance test A.3: `/req/global/sdu`

REQUIREMENT 1: INDIVIDUAL STANDARDIZATION TARGETS ARE INDEPENDENT

`/req/global/target-independence`

Included in	Conformance class A.1: `/conf/global`

There shall be no dependency between or among the individual Standardization Targets.

REQUIREMENT 2: IMPLEMENTATION CONFORMS TO THE LOGICAL MODEL

`/req/global/logical-model`

Included in	Conformance class A.1: `/conf/global`

Implementations of concrete data conforming to this standard SHALL conform to all dependent or inherited classes, attributes, and associations, multiplicities, and data types in the Logical Model.

REQUIREMENT 3: SDU CONFORMS TO THE "STRUCTURAL DATA UNIT — SDU" STEREOTYPE

`/req/global/sdu`

Included in	Conformance class A.1: `/conf/global`

Implementations using encoded SDUs SHALL conform to the logical description of the Logical Model elements with the "Structural Data Unit — SDU" stereotype.

8.3. Common requirements

8.3.1. Tangent point specification requirements

REQUIREMENTS CLASS 2: TANGENT POINT REQUIREMENTS

`/req/tangent-point`

Conformance test	Conformance class A.2: `/conf/tangent-point`

Common tangent point requirements for SDUs that include tangent points.

REQUIREMENTS CLASS 2: TANGENT POINT REQUIREMENTS

Requirement	Requirement 4: `/req/tangent-point/height`
Requirement	Requirement 5: `/req/tangent-point/latitude`
Requirement	Requirement 6: `/req/tangent-point/longitude`
Guidance	• The tangent plane `longitude`, `latitude`, and h parameters are specified without any conditions or constraints on precision to be used in an implementation. Any such constraints would be found as requirements on a specific implementation as an encoding.

REQUIREMENT 4: TANGENT POINT HEIGHT VALUE SPECIFICATION

`/req/tangent-point/height`

Included in	Requirements class 2: `/req/tangent-point`

An instance of a GeoPose `tangentPoint.h` attribute SHALL be expressed as a height in meters above the WGS-84 ellipsoid, represented as a signed as a signed floating point value conforming to IEEE 754. If the tangent point is above the WGS-84 ellipsoid, the value SHALL be positive. If the tangent point is below the WGS-84 ellipsoid, the value SHALL be negative.

REQUIREMENT 5: TANGENT POINT LATITUDE VALUE SPECIFICATION

`/req/tangent-point/latitude`

Included in	Requirements class 2: `/req/tangent-point`

An instance of GeoPose tangentPoint.latitude attribute SHALL be expressed as decimal degrees and represented as a signed floating point value conforming to IEEE 754. The minimum value shall be 90.0 degrees and the maximum value shall be 90.0 degrees.

REQUIREMENT 6: TANGENT POINT LONGITUDE VALUE SPECIFICATION

`/req/tangent-point/longitude`

Included in	Requirements class 2: `/req/tangent-point`

An instance of a GeoPose tangentPoint.longitude attribute SHALL be expressed as decimal degrees and represented as a signed floating point value conforming to IEEE 754. The minimum value shall be -180.0 degrees and the maximum value shall be 180.0 degrees.

8.3.2. Frame specification requirements

REQUIREMENTS CLASS 3: FRAME SPECIFICATION REQUIREMENTS

`/req/frame-spec`

Conformance test	Conformance class A.3: `/conf/frame-spec`

Common frame specification requirements for SDUs that include frames.

Requirement	Requirement 7: `/req/frame-spec/authority`
Requirement	Requirement 8: `/req/frame-spec/id`
Requirement	Requirement 9: `/req/frame-spec/parameters`

REQUIREMENT 7: FRAME SPECIFICATION AUTHORITY UNIQUELY SPECIFIES SOURCE OF REFERENCE FRAME SPECIFICATION

`/req/frame-spec/authority`

Included in	Requirements class 3: `/req/frame-spec`

The FrameSpecification.authority attribute SHALL contain a string uniquely specifying a source of reference frame specifications.

REQUIREMENT 8: FRAME SPECIFICATION ID UNIQUELY DEFINES FRAME WITHIN AUTHORITY

`/req/frame-spec/id`

Included in	Requirements class 3: `/req/frame-spec`

The FrameSpecification.ID attribute SHALL be a string uniquely defining a frame within the authority.

REQUIREMENT 9: FRAME SPECIFICATION PARAMETER CONTAINS ALL PARAMETERS NEEDED

`/req/frame-spec/parameters`

Included in	Requirements class 3: `/req/frame-spec`

REQUIREMENT 9: FRAME SPECIFICATION PARAMETER CONTAINS ALL PARAMETERS NEEDED

The FrameSpecification.parameter attribute SHALL contain all parameters needed for the corresponding authority and ID.

Guidance	• The definition of these parameters is outside the scope of GeoPose.

8.3.3. Time specification requirements

REQUIREMENTS CLASS 4: TIME SPECIFICATION REQUIREMENTS

`/req/time`

Conformance test	Conformance class A.4: `/conf/time`

Requirements on GeoPose time-related objects.

Requirement	Requirement 10: `/req/time/instant`
Requirement	Requirement 11: `/req/time/duration`

REQUIREMENT 10: IMPLEMENTATION OF GEOPOSE_INSTANT

`/req/time/instant`

Included in	Requirements class 4: `/req/time`

The `GeoPose_Instant` object shall express Unix Time in seconds multiplied by 1,000, with the unit of measure in milliseconds.

REQUIREMENT 11: IMPLEMENTATION OF GEOPOSE_DURATION

`/req/time/duration`

Included in	Requirements class 4: `/req/time`

The `GeoPose_Duration` object shall express time in seconds multiplied by 1,000, with the unit of measure in milliseconds.

8.4. SDU requirements

8.4.1. Requirements for Standardization Target 1: Basic-YPR

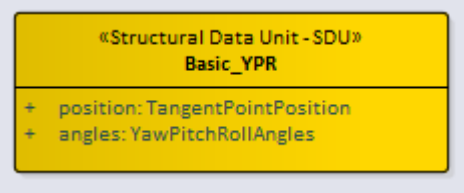

Figure 14 — Structure of the Basic YPR SDU

REQUIREMENTS CLASS 5: BASIC-YPR LOGICAL MODEL SDU	
`/req/basic-ypr`	
Conformance test	Conformance class A.5: `/conf/basic-ypr`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 2: `/req/tangent-point`
The Basic-YPR Target has a simple structure with no options. Position is specified as a point in an LTP-ENU frame and rotation is specified by yaw, pitch, and roll angles specified in decimal degrees.	
Requirement	Requirement 12: `/req/basic-ypr/position`
Requirement	Requirement 13: `/req/basic-ypr/angles`

REQUIREMENT 12: EXPRESSION OF OUTER FRAME	
`/req/basic-ypr/position`	
Included in	Requirements class 5: `/req/basic-ypr`
The `Basic_YPR.position` attribute shall represent the outer frame, specified by an implicit WGS-84 CRS and an implicit EPSG 4461-CS (LTP-ENU) coordinate system and explicit parameters to define the tangent point.	

REQUIREMENT 13: EXPRESSION OF INNER FRAME

`/req/basic-ypr/angles`

Included in	Requirements class 5: `/req/basic-ypr`

The `Basic_YPR.angles` attribute shall represent the inner frame, which is a rotation-only transformation with Yaw, Pitch, and Roll (YPR) angles, which expressed as three consecutive rotations of a reference frame oriented East-North-Up (ENU) coordinate system (where the coordinate axes East, North, and Up correspond to the axes X, Y, Z) about the local (rotated) axes z, y, and x, applied in that order, corresponding to the conventional Yaw, Pitch, and Roll angles. The unit of measure SHALL be the degree and the angles represented as signed real number values.

8.4.2. Requirements for Standardization Target 2: Basic-Quaternion

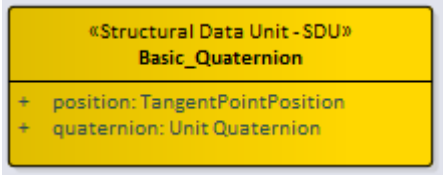

Figure 15 — Structure of the Basic Quaternion SDU

REQUIREMENTS CLASS 6: BASIC-QUATERNION LOGICAL MODEL SDU

`/req/basic-quaternion`

Conformance test	Conformance class A.6: `/conf/basic-quaternion`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 2: `/req/tangent-point`

The Basic-Quaternion Target has a simple structure with no options. Position is specified as a point in an LTP-ENU frame and rotation is specified as a unit quaternion.

Requirement	Requirement 14: `/req/basic-quaternion/position`
Requirement	Requirement 15: `/req/basic-quaternion/quaternion`

REQUIREMENT 14: EXPRESSION OF OUTER FRAME

`/req/basic-quaternion/position`

REQUIREMENT 14: EXPRESSION OF OUTER FRAME

Included in	Requirements class 6: `/req/basic-quaternion`

The `Basic_Quaternion.position` attribute shall represent the outer frame, specified by an implicit WGS-84 CRS and an implicit EPSG 4461-CS (LTP-ENU) coordinate system and explicit parameters to define the tangent point.

REQUIREMENT 15: EXPRESSION OF INNER FRAME

`/req/basic-quaternion/quaternion`

Included in	Requirements class 6: `/req/basic-quaternion`

The `Basic_Quaternion.quaternion` attribute shall represent the inner frame, which shall be a rotation-only transformation using a unit quaternion. The uniq quaternion shall be represented as an instance of a GeoPose Logical Model quaternion data type, expressed as four real numbers, representing four quaternion components w, x, y, z, in that sequential order. The sum of the squares of the individual components shall be as close to 1.0 as the real number representation allows. The quaternion shall be applied to an initial reference frame oriented East-North-Up (ENU) coordinate system where the coordinate axes East, North, and Up correspond to the axes X, Y, Z.

8.4.3. Requirements for Standardization Target 3: Advanced

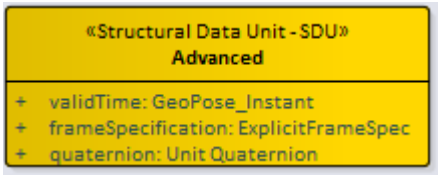

Figure 16 — Structure of the Basic Advanced SDU

REQUIREMENTS CLASS 7: ADVANCED LOGICAL MODEL SDU

`/req/advanced`

Conformance test	Conformance class A.7: `/conf/advanced`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 3: `/req/frame-spec`
Dependency	Requirements class 4: `/req/time`

The Advanced Target has a more general structure than Basic-YPR and Basic-Quaternion, supporting flexible specification of Outer Frame and a Valid Time.

REQUIREMENTS CLASS 7: ADVANCED LOGICAL MODEL SDU

Requirement	Requirement 16: `/req/advanced/valid-time`
Requirement	Requirement 17: `/req/advanced/frame-spec`
Requirement	Requirement 18: `/req/advanced/quaternion`

REQUIREMENT 16: EXPRESSION OF VALID TIME AS GEOPOSE_INSTANT

`/req/advanced/valid-time`

Included in	Requirements class 7: `/req/advanced`
Dependency	Requirement 10: `/req/time/instant`

The `Advanced.validTime` attribute shall be represented by a `GeoPose_Instant` object.

REQUIREMENT 17: EXPRESSION OF OUTER FRAME

`/req/advanced/frame-spec`

Included in	Requirements class 7: `/req/advanced`
Dependency	Requirements class 3: `/req/frame-spec`

The `Advanced.frameSpecification` attribute shall represent an explicit frame specification with the `ExplicitFrameSpec` object.

REQUIREMENT 18: EXPRESSION OF INNER FRAME

`/req/advanced/quaternion`

Included in	Requirements class 7: `/req/advanced`

The `Advanced.quaternion` attribute shall contain a quaternion expressed using the UnitQuaternion datatype value expressed as four real numbers, representing four quaternion components w, x, y, z, in that sequential order. The sum of the squares of the individual components SHALL be as close to 1.0 as the real number representation allows. The quaternion SHALL be applied to an initial reference frame oriented East-North-Up (ENU) coordinate system where the coordinate axes East, North, and Up correspond to the axes X, Y, Z.

8.4.4. Requirements for Standardization Target 4: Graph

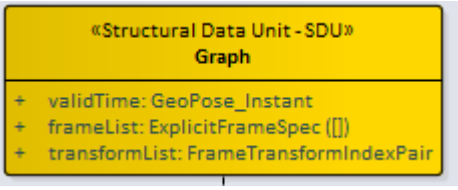

Figure 17 — Structure of the Graph SDU

REQUIREMENTS CLASS 8: GRAPH LOGICAL MODEL SDU	
`/req/graph`	
Conformance test	Conformance class A.8: `/conf/graph`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 3: `/req/frame-spec`
Dependency	Requirements class 4: `/req/time`
The Graph Target supports a network of object relative poses. The graph is a directed acyclic graph, each node must either be an Extrinsic Frame or reachable from an Extrinsic Frame.	
Requirement	Requirement 19: `/req/graph/valid-time`
Requirement	Requirement 20: `/req/graph/frame-list`
Requirement	Requirement 21: `/req/graph/transform-list`

REQUIREMENT 19: EXPRESSION OF VALID TIME AS GEOPOSE_INSTANT	
`/req/graph/valid-time`	
Included in	Requirements class 8: `/req/graph`
Dependency	Requirement 10: `/req/time/instant`
The `Graph.validTime` attribute shall be represented by a `GeoPose_Instant` object.	

REQUIREMENT 20: LIST OF FRAME SPECIFICATIONS

`/req/graph/frame-list`

Included in	Requirements class 8: `/req/graph`

The `Graph.frameList` attribute shall represent a list of explicit frame specifications with an array of `ExplicitFrameSpec` objects.

REQUIREMENT 21: TRANSFORMS FOR FRAME SPECIFICATION LIST

`/req/graph/transform-list`

Included in	Requirements class 8: `/req/graph`

The `Graph.transformList` attribute shall have a value of type `FrameTransformIndexPair`. Each index value in a `FrameListTransformPair` SHALL be a distinct integer value between 0 and one less than the number of elements in the `frameList` property.

8.4.5. Requirements for Standardization Target 5: Chain

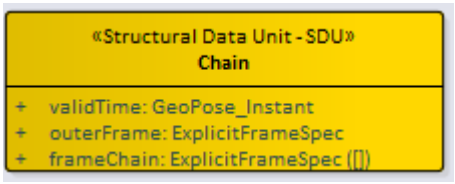

Figure 18 — Structure of the Chain SDU

REQUIREMENTS CLASS 9: CHAIN LOGICAL MODEL SDU

`/req/chain`

Conformance test	Conformance class A.9: `/conf/chain`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 3: `/req/frame-spec`
Dependency	Requirements class 4: `/req/time`

The Chain Target supports relationships between a linear sequence of pose relationships. The first frame in the sequence must be an Outer Frame.

REQUIREMENTS CLASS 9: CHAIN LOGICAL MODEL SDU

Requirement	Requirement 22: `/req/chain/valid-time`
Requirement	Requirement 23: `/req/chain/initial-frame`
Requirement	Requirement 24: `/req/chain/frame-chain`

REQUIREMENT 22: EXPRESSION OF VALID TIME AS GEOPOSE_INSTANT

`/req/chain/valid-time`

Included in	Requirements class 9: `/req/chain`
Dependency	Requirement 10: `/req/time/instant`

The `Chain.validTime` attribute shall be represented by a `GeoPose_Instant` object.

REQUIREMENT 23: SPECIFICATION OF INITIAL FRAME

`/req/chain/initial-frame`

Included in	Requirements class 9: `/req/chain`

The `Chain.outerFrame` attribute shall represent the first frame in the sequence with the `ExplicitFrameSpec` object.

REQUIREMENT 24: CHAIN OF FRAME SPECIFICATIONS

`/req/chain/frame-chain`

Included in	Requirements class 9: `/req/chain`

The `Chain.frameChain` attribute shall represent a list of explicit frame specifications with an array of `ExplicitFrameSpec` objects. Each index value shall be a distinct integer value between 0 and one less than the number of elements in the frameChain property.

8.4.6. Requirements for Standardization Target 6: Regular Series

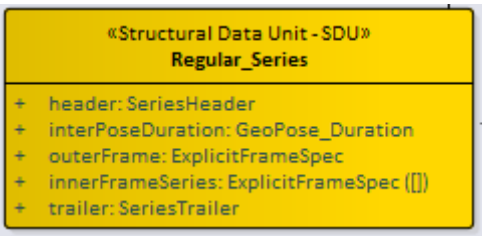

Figure 19 — Structure of the Regular Series SDU

REQUIREMENTS CLASS 10: REGULAR_SERIES LOGICAL MODEL SDU	
`/req/series-regular`	
Conformance test	Conformance class A.10: `/conf/series-regular`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 3: `/req/frame-spec`
Dependency	Requirements class 4: `/req/time`
The Regular_Series Target represents the time evolution of a single GeoPose, with a constant time duration between successive inner frames.	
Requirement	Requirement 25: `/req/series-regular/duration`
Requirement	Requirement 26: `/req/series-regular/outer-frame`
Requirement	Requirement 27: `/req/series-regular/inner-frame-series`
Requirement	Requirement 28: `/req/series-regular/header`
Requirement	Requirement 29: `/req/series-regular/trailer`

REQUIREMENT 25: EXPRESSION OF DURATION AS GEOPOSE_DURATION	
`/req/series-regular/duration`	
Included in	Requirements class 10: `/req/series-regular`

REQUIREMENT 25: EXPRESSION OF DURATION AS GEOPOSE_DURATION

Dependency	Requirement 11: /req/time/duration

The Regular_Series.interPoseDuration attribute shall be represented by an instance of the GeoPose_Duration object.

REQUIREMENT 26: EXPRESSION OF OUTER FRAME

/req/series-regular/outer-frame

Included in	Requirements class 10: /req/series-regular

The Regular_Series.outerFrame attribute shall represent the first frame in the series with the ExplicitFrameSpec object.

REQUIREMENT 27: EXPRESSION OF INNER FRAMES

/req/series-regular/inner-frame-series

Included in	Requirements class 10: /req/series-regular

The Regular_Series.innerFrameSeries attribute shall represent the succession of inner frames as an array of ExplicitFrameSpec objects.

REQUIREMENT 28: EXPRESSION OF SERIES HEADER

/req/series-regular/header

Included in	Requirements class 10: /req/series-regular

The Regular_Series.header attribute shall be implemented as an instance of SeriesHeader.

REQUIREMENT 29: EXPRESSION OF SERIES TRAILER

/req/series-regular/trailer

Included in	Requirements class 10: /req/series-regular

The Regular_Series.trailer attribute shall be implemented as an instance of SeriesTrailer.

8.4.7. Requirements for Standardization Target 7: Irregular Series

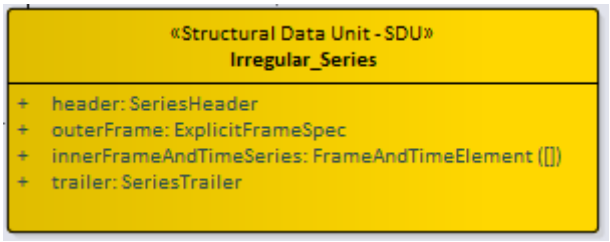

Figure 20 — Structure of the Irregular Series SDU

REQUIREMENTS CLASS 11: IRREGULAR_SERIES LOGICAL MODEL SDU	
`/req/series-irregular`	
Conformance test	Conformance class A.11: `/conf/series-irregular`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 3: `/req/frame-spec`
Dependency	Requirements class 4: `/req/time`
The Irregular_Series Target represents the time evolution of a single GeoPose, with a variable time duration between successive inner frames.	
Requirement	Requirement 30: `/req/series-irregular/inner-frame-and-time`
Requirement	Requirement 31: `/req/series-irregular/outer-frame`
Requirement	Requirement 32: `/req/series-irregular/header`
Requirement	Requirement 33: `/req/series-irregular/trailer`

REQUIREMENT 30: EXPRESSION OF INNER FRAMES AND TIME SERIES	
`/req/series-irregular/inner-frame-and-time`	
Included in	Requirements class 11: `/req/series-irregular`
Dependency	Requirement 10: `/req/time/instant`

REQUIREMENT 30: EXPRESSION OF INNER FRAMES AND TIME SERIES

The `Irregular_Series.innerFrameAndTime` attribute SHALL be implemented as an array of `FrameAndTimeElement` objects, each of which is a pair of `ExplicitFrameSpec` and `GeoPoseInstant` objects.

REQUIREMENT 31: EXPRESSION OF OUTER FRAME

`/req/series-irregular/outer-frame`

Included in	Requirements class 11: `/req/series-irregular`

The `Irregular_Series.outerFrame` attribute shall represent the first frame in the series expressed by the `innerFrameAndTime` attribute.

REQUIREMENT 32: EXPRESSION OF SERIES HEADER

`/req/series-irregular/header`

Included in	Requirements class 11: `/req/series-irregular`

The `Irregular_Series.header` attribute shall be implemented as an instance of SeriesHeader.

REQUIREMENT 33: EXPRESSION OF SERIES TRAILER

`/req/series-irregular/trailer`

Included in	Requirements class 11: `/req/series-irregular`

The `Irregular_Series.trailer` attribute shall be implemented as an instance of SeriesTrailer.

8.4.8. Requirements for Standardization Target 8: Stream

The Stream target consists of two parts: a single initial specification of a transition model and an outer frame (the Stream Header) and zero or more time-stamped frame specifications (the Stream Elements). In the delivery of a stream the Header and Elements are not part of a single data structure that exists at a single instant.

Nevertheless, recording the Header and all of the Elements received up to some point in time in a single structure is possible. The result is that there are two kinds of data objects that may be involved in transmission of a stream: Headers and Elements and a third kind of object that represents a Recorded Stream.

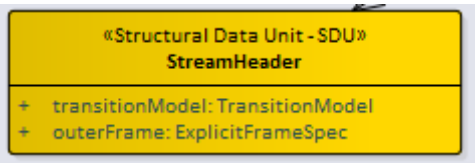

Figure 21 — Structure of the Stream Header SDU

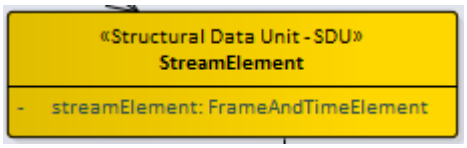

Figure 22 — Structure of the Stream Element SDU

REQUIREMENTS CLASS 12: STREAMHEADER AND STREAMELEMENT LOGICAL MODEL SDUS

`/req/stream`

Conformance test	Conformance class A.12: `/conf/stream`
Dependency	Requirements class 1: `/req/global`
Dependency	Requirements class 3: `/req/frame-spec`

The Stream target supports a network of object relative poses. The graph is a directed acyclic graph, each node must either be an Extrinsic Frame or reachable from an Extrinsic Frame.

Requirement	Requirement 34: `/req/stream/header-initial-frame`
Requirement	Requirement 35: `/req/stream/header-transition-model`
Requirement	Requirement 36: `/req/stream/element`

REQUIREMENT 34: EXPRESSION OF OUTER FRAME IN STREAMHEADER

`/req/stream/header-initial-frame`

Included in	Requirements class 12: `/req/stream`

The `StreamHeader.outerFrame` attribute shall represent the initial frame of the stream with a value that is an instant of the `ExplicitFrameSpec` object.

REQUIREMENT 35: EXPRESSION OF TRANSITION MODEL IN STREAMHEADER

`/req/stream/header-transition-model`

Included in	Requirements class 12: `/req/stream`

The `StreamHeader.transitionModel` attribute shall have a value that is an instance of the `TransitionModel` enumeration.

REQUIREMENT 36: EXPRESSION OF STREAM ELEMENTS IN STREAMELEMENT

`/req/stream/element`

Included in	Requirements class 12: `/req/stream`
Dependency	Requirement 10: `/req/time/instant`

The `StreamElement.streamElement` attribute shall be implemented as an array of `FrameAndTimeElement` objects, each of which is a pair of `ExplicitFrameSpec` and `GeoPoseInstant` objects.

9

REQUIREMENTS FOR ENCODINGS

9 REQUIREMENTS FOR ENCODINGS

9.1. General

Requirements Classes are modularized based on the corresponding Standardization Target. This results in some SDU requirements being repeated between Targets. SDU requirements are abstract in the sense that SDUs are implemented as concrete data objects via serialization formats or encodings. Therefore, there are additional requirements that specify how each Target's group of SDUs are encoded. If there are multiple encodings of a Target, then there is a corresponding additional set of encoding requirements in the Target's section. This occurs only once in GeoPose 1.0, with two different levels of JSON encoding strictness individually specified for the Basic-Q Target.

9.2. JSON Encoding

9.2.1. General

The JSON encoding is one of many possible ways of implementing a concrete representation of any one or more of the GeoPose Standardization Targets. The specific JSON encoding in this section follows all of the normal rules and conventions of JSON. For example, the order of named properties is not significant and, except in the case of the "strict" Basic-Quaternion object, there is no restriction on adding additional properties not specified in the GeoPose 1.0 Standard. In addition to supporting specific application requirements outside the GeoPose 1.0 scope, this flexibility enables experimentation with useful properties that might be part of a future version of the OGC GeoPose Standard.

9.2.2. Standardization Target 1: Basic-YPR

REQUIREMENTS CLASS 13: JSON ENCODING OF BASIC-YPR SDU

`/req/basic-ypr-encoding-json`

Conformance test	Conformance class A.13: `/conf/basic-ypr-encoding-json`

Requirements for the JSON encoding of a Basic-YPR SDU.

REQUIREMENTS CLASS 13: JSON ENCODING OF BASIC-YPR SDU

Requirement	Requirement 37: `/req/basic-ypr-encoding-json/definition`

REQUIREMENT 37: SPECIFICATION AS JSON SCHEMA

`/req/basic-ypr-encoding-json/definition`

Included in	Requirements class 13: `/req/basic-ypr-encoding-json`

A JSON-encoded Basic-YPR GeoPose SHALL conform to the Basic-YPR JSON-Schema 2019-9 definition (Figure 23).

Guidance	• This JSON encoding is extensible because the JSON-Schema "additional Properties" property is set to the default value of **true**.

```
{
  "description": "Basic-YPR: Basic GeoPose using yaw, pitch, and roll to
specify orientation",
  "definitions": {
    "angles": {
      "type": "object",
      "properties": {
        "yaw": {
          "type": "number"
        },
        "pitch": {
          "type": "number"
        },
        "roll": {
          "type": "number"
        }
      },
      "required": [
        "yaw",
        "pitch",
        "roll"
      ]
    },
    "Position": {
      "type": "object",
      "properties": {
        "lat": {
          "type": "number"
        },
        "lon": {
          "type": "number"
        },
        "h": {
          "type": "number"
        }
      },
      "required": [
        "lat",
        "lon",
```

```
        "h"
      ]
    }
  },
  "type": "object",
  "properties": {
    "position": {
      "$ref": "#/definitions/Position"
    },
    "angles": {
      "$ref": "#/definitions/angles"
    }
  },
  "required": [
    "position",
    "angles"
  ]
}
```

Figure 23 — Basic-YPR: JSON encoding schema

Example — Basic-YPR: JSON encoding example

```
{
  "position": {
    "lat": 47.7,
    "lon": -122.3,
    "h": 11.5
  },
  "angles": {
    "yaw": 5.514456741060452,
    "pitch": -0.43610515937237904,
    "roll": 0.0
  }
}
```

9.2.3. Standardization Target 2: Basic-Quaternion

Two JSON encodings are defined for the Basic-Quaternion Target:

Strict disallowing additional JSON properties not defined in the schema

Extensible allowing additional JSON properties in addition to those required by the schema.

All other targets follow the default and permit additional JSON properties.

9.2.3.1. Strict JSON Encoding

REQUIREMENTS CLASS 14: STRICT JSON ENCODING OF BASIC-QUATERNION SDU

`/req/basic-quaternion-encoding-json-strict`

REQUIREMENTS CLASS 14: STRICT JSON ENCODING OF BASIC-QUATERNION SDU

Conformance test	Conformance class A.14: `/conf/basic-quaternion-encoding-json-strict`

Requirements for the JSON encoding of a Basic-Quaternion SDU in strict mode. The strict encoding does not allow specification of JSON elements outside of the schema.

Requirement	Requirement 38: `/req/basic-quaternion-encoding-json-strict/definition`

REQUIREMENT 38: SPECIFICATION AS JSON SCHEMA

`/req/basic-quaternion-encoding-json-strict/definition`

Included in	Requirements class 14: `/req/basic-quaternion-encoding-json-strict`

A JSON-encoded Basic-Quaternion GeoPose SHALL conform to the strict Basic-Quaternion JSON-Schema 2019-9 definition (Figure 24).

```
{
  "description": "Basic-Quaternion-Strict: Basic GeoPose using quaternion to
specify orientation - no additional properties",
  "definitions": {
    "Position": {
      "type": "object",
      "properties": {
        "lat": {
          "type": "number"
        },
        "lon": {
          "type": "number"
        },
        "h": {
          "type": "number"
        }
      },
      "required": [
        "lat",
        "lon",
        "h"
      ]
    },
    "Quaternion": {
      "type": "object",
      "properties": {
        "x": {
          "type": "number"
        },
        "y": {
          "type": "number"
        },
        "z": {
          "type": "number"
        },
```

```
        "w": {
          "type": "number"
        }
      },
      "required": [
        "x",
        "y",
        "z",
        "w"
      ]
    }
  },
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "position": {
      "$ref": "#/definitions/Position"
    },
    "quaternion": {
      "$ref": "#/definitions/Quaternion"
    }
  },
  "required": [
    "position",
    "quaternion"
  ]
}
```

Figure 24 — Basic-Quaternion: Strict JSON encoding schema

Example

```
{
  "position": {
    "lat": 47.7,
    "lon": -122.3,
    "h": 11.5
  },
  "quaternion": {
    "x": 0.20054473382601948,
    "y": -0.08111675703887213,
    "z": 0.3660908114262869,
    "w": -0.9050852994339209
  }
}
```

9.2.3.2. Permissive JSON Encoding

This JSON encoding of the Basic-Quaternion GeoPose is extensible because the JSON-Schema "additionalProperties" property is set to the default value of **true**. This encoding is intended to be the default GeoPose.

REQUIREMENTS CLASS 15: PERMISSIVE JSON ENCODING OF BASIC-QUATERNION SDU

`/req/basic-quaternion-encoding-json`

REQUIREMENTS CLASS 15: PERMISSIVE JSON ENCODING OF BASIC-QUATERNION SDU

Conformance test	Conformance class A.15: `/conf/basic-quaternion-encoding-json`

Requirements for the JSON encoding of a Basic-Quaternion SDU.

Requirement	Requirement 39: `/req/basic-quaternion-encoding-json/definition`

REQUIREMENT 39: SPECIFICATION AS JSON SCHEMA

`/req/basic-quaternion-encoding-json/definition`

Included in	Requirements class 15: `/req/basic-quaternion-encoding-json`

A JSON-encoded Basic-Quaternion GeoPose SHALL conform to the Basic-Quaternion JSON-Schema 2019-9 definition (Figure 25).

```
{
  "description": "Basic-Quaternion-Strict: Basic GeoPose using quaternion to
specify orientation - no additional properties",
  "definitions": {
    "Position": {
      "type": "object",
      "properties": {
        "lat": {
          "type": "number"
        },
        "lon": {
          "type": "number"
        },
        "h": {
          "type": "number"
        }
      },
      "required": [
        "lat",
        "lon",
        "h"
      ]
    },
    "Quaternion": {
      "type": "object",
      "properties": {
        "x": {
          "type": "number"
        },
        "y": {
          "type": "number"
        },
        "z": {
          "type": "number"
        },
```

```
        "w": {
          "type": "number"
        }
      },
      "required": [
        "x",
        "y",
        "z",
        "w"
      ]
    }
  },
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "position": {
      "$ref": "#/definitions/Position"
    },
    "quaternion": {
      "$ref": "#/definitions/Quaternion"
    }
  },
  "required": [
    "position",
    "quaternion"
  ]
}
```

Figure 25 — Basic-Quaternion: Permissive JSON encoding schema

Example — Basic-Quaternion: Permissive JSON encoding example

```
{
  "position": {
    "lat": 47.7,
    "lon": -122.3,
    "h": 11.5
  },
  "quaternion": {
    "x": 0.20054473382601948,
    "y": -0.08111675703887213,
    "z": 0.3660908114262869,
    "w": -0.9050852994339209
  }
}
```

9.2.4. Standardization Target 3: Advanced GeoPose

REQUIREMENTS CLASS 16: JSON ENCODING OF ADVANCED SDU

`/req/advanced-encoding-json`

Conformance test	Conformance class A.16: `/conf/advanced-encoding-json`

Requirements for the JSON encoding of a Advanced SDU.

REQUIREMENTS CLASS 16: JSON ENCODING OF ADVANCED SDU

Requirement	Requirement 40: `/req/advanced-encoding-json/definition`

REQUIREMENT 40: SPECIFICATION AS JSON SCHEMA

`/req/advanced-encoding-json/definition`

Included in	Requirements class 16: `/req/advanced-encoding-json`

A JSON-encoded Advanced GeoPose SHALL conform to the Advanced JSON-Schema 2019-9 definition (Figure 26).

```
{
  "description": "Advanced: Advanced GeoPose allowing flexible outer frame
specification, quaternion orientation, and valid time.",
  "definitions": {
    "FrameSpecification": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    },
    "Quaternion": {
      "type": "object",
      "properties": {
        "x": {
          "type": "number"
        },
        "y": {
          "type": "number"
        },
        "z": {
          "type": "number"
        },
        "w": {
          "type": "number"
        }
      },
      "required": [
        "x",
        "y",
```

```
        "z",
        "w"
      ]
    }
  },
  "type": "object",
  "properties": {
    "frameSpecification": {
      "$ref": "#/definitions/FrameSpecification"
    },
    "quaternion": {
      "$ref": "#/definitions/Quaternion"
    },
    "validTime": {
      "type": "integer"
    }
  },
  "required": [
    "frameSpecification",
    "quaternion"
  ]
}
```

Figure 26 — Advanced GeoPose: JSON encoding schema

Example — Advanced GeoPose: JSON encoding example

```
{
  "frameSpecification": {
    "authority": "/geopose/1.0",
    "id": "LTP-ENU",
    "parameters": "longitude=-122.3000000&latitude=47.7000000&height=11.000"
  },
  "quaternion": {
    "x": 0.20056154657066608,
    "y": -0.08111602541464237,
    "z": 0.36606032744426537,
    "w": -0.9050939692261301
  },
  "validTime": 1630560671227
}
```

9.2.5. Standardization Target 4: Graph

REQUIREMENTS CLASS 17: JSON ENCODING OF GRAPH SDU	
/req/graph-encoding-json	
Conformance test	Conformance class A.17: /conf/graph-encoding-json
Requirements for the JSON encoding of a Graph SDU.	
Requirement	Requirement 41: /req/graph-encoding-json/definition

REQUIREMENT 41: SPECIFICATION AS JSON SCHEMA

`/req/graph-encoding-json/definition`

Included in	Requirements class 17: `/req/graph-encoding-json`

A JSON-encoded GeoPose Graph SHALL conform to the GeoPose Chain JSON-Schema 2019-9 definition (Figure 27).

Guidance	• This JSON encoding is extensible because the JSON-Schema "additional Properties" property is set to the default value of **true**.

```
{
  "description": "Graph: An general structure modelling the pose relationship
between frames (nodes) and transforms (edges) in a graph structure.",
  "definitions": {
    "FrameSpecification": {
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    },
    "FrameTransformPair": {
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "link": {
          "type": [
            "array",
            "null"
          ],
          "items": {
            "type": "integer"
          }
        }
      },
      "required": [
        "link"
      ]
    }
```

```
  },
  "type": "object",
  "properties": {
    "validTime": {
      "type": "integer"
    },
    "frameList": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/FrameSpecification"
      },
      "minItems": 2
    },
    "transformList": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/FrameTransformPair"
      },
      "minItems": 1
    }
  },
  "required": [
    "validTime",
    "frameList",
    "transformList"
  ]
}
```

Figure 27 — Graph: JSON encoding schema

Example — Graph: JSON encoding example

```
{
  "validTime": 1630560671313,
  "frameList": [
    {
      "authority": "/geopose/1.0",
      "id": "/Extrinsic/LTP-ENU",
      "parameters": "longitude=-122.3000000&latitude=47.7000000&height=11.000"
    },
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    },
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    },
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    },
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
```

```
    },
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    },
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    }
  ],
  "transformList": [
    {
      "link": [
        0,
        1
      ]
    },
    {
      "link": [
        1,
        2
      ]
    },
    {
      "link": [
        2,
        3
      ]
    },
    {
      "link": [
        0,
        4
      ]
    },
    {
      "link": [
        4,
        5
      ]
    },
    {
      "link": [
        5,
        6
      ]
    }
  ]
}
```


9.2.6. Standardization Target 5: Chain

REQUIREMENTS CLASS 18: JSON ENCODING OF CHAIN SDU

/req/chain-encoding-json

Conformance test	Conformance class A.18: /conf/chain-encoding-json

Requirements for the JSON encoding of a Chain SDU.

Requirement	Requirement 42: /req/chain-encoding-json/definition

REQUIREMENT 42: SPECIFICATION AS JSON SCHEMA

/req/chain-encoding-json/definition

Included in	Requirements class 18: /req/chain-encoding-json

A JSON-encoded GeoPose Chain SHALL conform to the GeoPose Chain JSON-Schema 2019-9 definition (Figure 28).

Guidance	• This JSON encoding is extensible because the JSON-Schema "additional Properties" property is set to the default value of **true**.

```
{
  "description": "Chain: An outer frame and a sequence of transformations to a
final innermost frame.",
  "definitions": {
    "FrameSpecification": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    },
    "FrameSpecification-1": {
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "authority": {
```

```
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    }
  },
  "type": "object",
  "properties": {
    "validTime": {
      "type": "integer"
    },
    "outerFrame": {
      "$ref": "#/definitions/FrameSpecification"
    },
    "frameChain": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/FrameSpecification-1"
      },
      "minItems": 2
    }
  },
  "required": [
    "validTime",
    "outerFrame",
    "frameChain"
  ]
}
```

Figure 28 — Chain: JSON encoding schema

Example — Chain: JSON encoding example

```
{
  "validTime": 1630560671263,
  "outerFrame": {
    "authority": "/geopose/1.0",
    "id": "/Extrinsic/LTP-ENU",
    "parameters": "longitude=-122.3000000&latitude=47.7000000&height=11.000"
  },
  "frameChain": [
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    },
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    },
```

```
    {
      "authority": "/geopose/1.0",
      "id": "/Intrinsic/Translate-Rotate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[0.69291, 0.69291, 0.
14097, 0.14097]"
    }
  ]
}
```

9.2.7. Standardization Target 6: Regular Series

REQUIREMENTS CLASS 19: JSON ENCODING OF REGULAR SERIES SDU

`/req/series-regular-encoding-json`

Conformance test	Conformance class A.19: `/conf/series-regular-encoding-json`

Requirements for the JSON encoding of a Regular Series SDU.

Requirement	Requirement 43: `/req/series-regular-encoding-json/definition`

REQUIREMENT 43: SPECIFICATION AS JSON SCHEMA

`/req/series-regular-encoding-json/definition`

Included in	Requirements class 19: `/req/series-regular-encoding-json`

A JSON-encoded GeoPose Regular Series SHALL conform to the GeoPose Chain JSON-Schema 2019-9 definition (Figure 29).

Guidance	• This JSON encoding is extensible because the JSON-Schema “additional Properties” property is set to the default value of **true**.

```
{
  "description": "Regular Series: Regular GeoPose time series with constant
inter-pose duration.",
  "definitions": {
    "FrameSpecification": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
```

```
      }
    },
    "required": [
      "authority",
      "id",
      "parameters"
    ]
  },
  "FrameSpecification-1": {
    "type": [
      "object",
      "null"
    ],
    "properties": {
      "authority": {
        "type": "string"
      },
      "id": {
        "type": "string"
      },
      "parameters": {
        "type": "string"
      }
    },
    "required": [
      "authority",
      "id",
      "parameters"
    ]
  },
  "SeriesHeader": {
    "type": "object",
    "properties": {
      "poseCount": {
        "type": "integer"
      },
      "integrityCheck": {
        "type": [
          "string",
          "null"
        ]
      },
      "startInstant": {
        "type": "integer"
      },
      "stopInstant": {
        "type": "integer"
      },
      "transitionModel": {
        "$ref": "#/definitions/TransitionModel"
      }
    },
    "required": [
      "poseCount",
      "startInstant",
      "stopInstant",
      "transitionModel"
    ]
  },
  "SeriesTrailer": {
    "type": "object",
    "properties": {
      "poseCount": {
```

```
          "type": "integer"
        },
        "integrityCheck": {
          "type": [
            "string",
            "null"
          ]
        }
      },
      "required": [
        "poseCount"
      ]
    },
    "TransitionModel": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    }
  },
  "type": "object",
  "properties": {
    "header": {
      "$ref": "#/definitions/SeriesHeader"
    },
    "interPoseDuration": {
      "type": "integer"
    },
    "outerFrame": {
      "$ref": "#/definitions/FrameSpecification"
    },
    "innerFrameSeries": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/FrameSpecification-1"
      },
      "minItems": 1
    },
    "trailer": {
      "$ref": "#/definitions/SeriesTrailer"
    }
  },
  "required": [
    "header",
    "interPoseDuration",
    "outerFrame",
    "innerFrameSeries",
    "trailer"
  ]
```

```
}
```

Figure 29 — Regular series: JSON encoding schema

Example — Regular series: JSON encoding example


```
{
  "header": {
    "poseCount": 2,
    "integrityCheck": "{\"SHA256\": \"5556fb65f8bf9eddb3ace1329c9a6aeedd4833409
965aeee3e6b61ed21f47858\"}",
    "startInstant": 1630560671367,
    "stopInstant": 1630560716367,
    "transitionModel": {
      "authority": "/geopose/1.0",
      "id": "none",
      "parameters": ""
    }
  },
  "interPoseDuration": 1000,
  "outerFrame": {
    "authority": "/geopose/1.0",
    "id": "LTP-ENU",
    "parameters": "longitude=-122.3000000&latitude=47.7000000&height=11.000"
  },
  "innerFrameSeries": [
    {
      "authority": "/geopose/1.0",
      "id": "RotateTranslate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[1.0, 0.0, 0.0, 0.5]"
    },
    {
      "authority": "/geopose/1.0",
      "id": "RotateTranslate",
      "parameters": "translation=[0.5, 0.0, 0.0]&rotation=[1.0, 0.0, 0.0, 0.5]"
    },
    {
      "authority": "/geopose/1.0",
      "id": "RotateTranslate",
      "parameters": "translation=[1.0, 0.0, 0.0]&rotation=[1.0, 0.0, 0.0, 0.5]"
    }
  ],
  "trailer": {
    "poseCount": 2,
    "integrityCheck": "{\"SHA256\": \"5556fb65f8bf9eddb3ace1329c9a6aeedd4833409
965aeee3e6b61ed21f47858\"}"
  }
}
```


9.2.8. Standardization Target 7: Irregular Series

REQUIREMENTS CLASS 20: JSON ENCODING OF IRREGULAR SERIES SDU

`/req/series-irregular-encoding-json`

Conformance test	Conformance class A.20: `/conf/series-irregular-encoding-json`

REQUIREMENTS CLASS 20: JSON ENCODING OF IRREGULAR SERIES SDU

Requirements for the JSON encoding of a Irregular Series SDU.

Requirement	Requirement 44: `/req/series-irregular-encoding-json/definition`

REQUIREMENT 44: SPECIFICATION AS JSON SCHEMA

`/req/series-irregular-encoding-json/definition`

Included in	Requirements class 20: `/req/series-irregular-encoding-json`

A JSON-encoded GeoPose Irregular Series SHALL conform to the GeoPose Chain JSON-Schema 2019-9 definition (Figure 30).

Guidance	• This JSON encoding is extensible because the JSON-Schema "additional Properties" property is set to the default value of **true**.

```
{
  "description": "Irregular Series: Irregular GeoPose time series with
variable inter-pose duration.",
  "definitions": {
    "FrameAndTime": {
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "frame": {
          "$ref": "#/definitions/FrameSpecification"
        },
        "validTime": {
          "type": "integer"
        }
      },
      "required": [
        "frame"
      ]
    },
    "FrameSpecification": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
```

```
      "id",
      "parameters"
    ]
  },
  "SeriesHeader": {
    "type": "object",
    "properties": {
      "poseCount": {
        "type": "integer"
      },
      "integrityCheck": {
        "type": [
          "string",
          "null"
        ]
      },
      "startInstant": {
        "type": "integer"
      },
      "stopInstant": {
        "type": "integer"
      },
      "transitionModel": {
        "$ref": "#/definitions/TransitionModel"
      }
    },
    "required": [
      "poseCount",
      "startInstant",
      "stopInstant",
      "transitionModel"
    ]
  },
  "SeriesTrailer": {
    "type": "object",
    "properties": {
      "poseCount": {
        "type": "integer"
      },
      "integrityCheck": {
        "type": [
          "string",
          "null"
        ]
      }
    },
    "required": [
      "poseCount"
    ]
  },
  "TransitionModel": {
    "type": "object",
    "properties": {
      "authority": {
        "type": "string"
      },
      "id": {
        "type": "string"
      },
      "parameters": {
        "type": "string"
      }
    },
```

```
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    }
  },
  "type": "object",
  "properties": {
    "header": {
      "$ref": "#/definitions/SeriesHeader"
    },
    "outerFrame": {
      "$ref": "#/definitions/FrameSpecification"
    },
    "innerFrameAndTimeSeries": {
      "type": "array",
      "items": {
        "$ref": "#/definitions/FrameAndTime"
      },
      "minItems": 1
    },
    "trailer": {
      "$ref": "#/definitions/SeriesTrailer"
    }
  },
  "required": [
    "header",
    "outerFrame",
    "innerFrameAndTimeSeries",
    "trailer"
  ]
}
```

Figure 30 — Irregular series: JSON encoding schema

Example — Irregular series: JSON encoding example

```
{
  "header": {
    "poseCount": 2,
    "integrityCheck": "{\"SHA256\": \"5556fb65f8bf9eddb3ace1329c9a6aeedd4833409
965aeee3e6b61ed21f47858\"}",
    "startInstant": 1630560671429,
    "stopInstant": 1630560716429,
    "transitionModel": {
      "authority": "/geopose/1.0",
      "id": "none",
      "parameters": ""
    }
  },
  "outerFrame": {
    "authority": "/geopose/1.0",
    "id": "LTP-ENU",
    "parameters": "longitude=-122.3000000&latitude=47.7000000&height=11.000"
  },
  "innerFrameAndTimeSeries": [
    {
      "frame": {
        "authority": "/geopose/1.0",
        "id": "RotateTranslate",
        "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[1.0, 0.0, 0.0, 0.
5]"
```

```
      },
      "validTime": 1630560671429
    },
    {
      "frame": {
        "authority": "/geopose/1.0",
        "id": "RotateTranslate",
        "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[1.0, 0.0, 0.0, 0.
5]"
      },
      "validTime": 1630560671429
    },
    {
      "frame": {
        "authority": "/geopose/1.0",
        "id": "RotateTranslate",
        "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[1.0, 0.0, 0.0, 0.
5]"
      },
      "validTime": 1630560671429
    }
  ],
  "trailer": {
    "poseCount": 2,
    "integrityCheck": "{\"SHA256\": \"5556fb65f8bf9eddb3ace1329c9a6aeedd4833409
965aeee3e6b61ed21f47858\"}"
  }
}
```


9.2.9. Standardization Target 8: Stream

REQUIREMENTS CLASS 21: JSON ENCODING OF STREAM SDUS

`/req/stream-encoding-json`

Conformance test	Conformance class A.21: `/conf/stream-encoding-json`

Requirements for the JSON encoding of Stream SDUs.

Requirement	Requirement 45: `/req/stream-encoding-json/element`
Requirement	Requirement 46: `/req/stream-encoding-json/header`
Requirement	Requirement 47: `/req/stream-encoding-json/record`

REQUIREMENT 45: STREAM ELEMENT SPECIFICATION AS JSON SCHEMA

`/req/stream-encoding-json/element`

REQUIREMENT 45: STREAM ELEMENT SPECIFICATION AS JSON SCHEMA

Included in	Requirements class 21: `/req/stream-encoding-json`

A JSON-encoded GeoPose Stream Element SHALL conform to the GeoPose Stream Element JSON-Schema 2019-9 definition (Figure 32).

REQUIREMENT 46: STREAM HEADER SPECIFICATION AS JSON SCHEMA

`/req/stream-encoding-json/header`

Included in	Requirements class 21: `/req/stream-encoding-json`

A JSON-encoded GeoPose Stream Element SHALL conform to the GeoPose Stream Header JSON-Schema 2019-9 definition (Figure 31).

Guidance	• This JSON encoding is extensible because the JSON-Schema "additional Properties" property is set to the default value of **true**.

REQUIREMENT 47: STREAM RECORD SPECIFICATION AS JSON SCHEMA

`/req/stream-encoding-json/record`

Included in	Requirements class 21: `/req/stream-encoding-json`

A JSON-encoded GeoPose Stream Record (a recorded Stream) SHALL conform to the GeoPose Stream Record JSON-Schema 2019-9 definition (Figure 33).

```
{
  "description": "Composite: StreamHeader – appears once at the beginning of a
stream.",
  "definitions": {
    "FrameSpecification": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
```

```
    },
    "TransitionModel": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    }
  },
  "type": "object",
  "properties": {
    "transitionModel": {
      "$ref": "#/definitions/TransitionModel"
    },
    "outerFrame": {
      "$ref": "#/definitions/FrameSpecification"
    }
  },
  "required": [
    "transitionModel",
    "outerFrame"
  ]
}
```

Figure 31 — Stream header: JSON encoding schema

```
{
  "description": "Stream Element: The repeated information streamed at
irregular times.",
  "definitions": {
    "FrameAndTime": {
      "type": "object",
      "properties": {
        "frame": {
          "$ref": "#/definitions/FrameSpecification"
        },
        "validTime": {
          "type": "integer"
        }
      },
      "required": [
        "frame"
      ]
    },
    "FrameSpecification": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
```

```
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    }
  },
  "type": "object",
  "properties": {
    "streamElement": {
      "$ref": "#/definitions/FrameAndTime"
    }
  },
  "required": [
    "streamElement"
  ]
}
```

Figure 32 — Stream element: JSON encoding schema

Refer to the specification of Requirement /req/stream-encoding-json/record.

```
{
  "description": "Stream: GeoPose stream is an open-ended irregular timeseries
suitable for streaming from a sensor or information service.",
  "definitions": {
    "FrameAndTime": {
      "type": "object",
      "properties": {
        "frame": {
          "$ref": "#/definitions/FrameSpecification"
        },
        "validTime": {
          "type": "integer"
        }
      },
      "required": [
        "frame"
      ]
    },
    "FrameSpecification": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
```

```
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    },
    "StreamElement": {
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "streamElement": {
          "$ref": "#/definitions/FrameAndTime"
        }
      },
      "required": [
        "streamElement"
      ]
    },
    "StreamHeader": {
      "type": "object",
      "properties": {
        "transitionModel": {
          "$ref": "#/definitions/TransitionModel"
        },
        "outerFrame": {
          "$ref": "#/definitions/FrameSpecification"
        }
      },
      "required": [
        "transitionModel",
        "outerFrame"
      ]
    },
    "TransitionModel": {
      "type": "object",
      "properties": {
        "authority": {
          "type": "string"
        },
        "id": {
          "type": "string"
        },
        "parameters": {
          "type": "string"
        }
      },
      "required": [
        "authority",
        "id",
        "parameters"
      ]
    }
  },
  "type": "object",
  "properties": {
    "header": {
      "$ref": "#/definitions/StreamHeader"
    },
    "streamElements": {
      "type": [
        "array",
```

```
        "null"
      ],
      "items": {
        "$ref": "#/definitions/StreamElement"
      },
      "minItems": 1
    }
  },
  "required": [
    "header",
    "streamElements"
  ]
}
```

Figure 33 — Stream record: JSON encoding schema

Example 1 — Valid JSON encoding of a Stream Header instance

```
{
  "transitionModel": {
    "authority": "/geopose/1.0",
    "id": "interpolate",
    "parameters": ""
  },
  "outerFrame": {
    "authority": "/geopose/1.0",
    "id": "LTP-ENU",
    "parameters": "longitude=-122.3000000&latitude=47.7000000&height=11.000"
  }
}
```

Example 2 — Valid JSON encoding of a Stream Element instance

```
{
  "streamElement": {
    "frame": {
      "authority": "/geopose/1.0",
      "id": "RotateTranslate",
      "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[-0.90510, 0.20057, -
0.08112, 0.36605]"
    },
    "validTime": 1630560671474
  }
}
```

Example 3 — Valid JSON encoding of a Recorded Stream

```
{
  "header": {
    "transitionModel": {
      "authority": "/geopose/1.0",
      "id": "interpolate",
      "parameters": ""
    },
    "outerFrame": {
      "authority": "/geopose/1.0",
      "id": "LTP-ENU",
      "parameters": "longitude=-122.3000000&latitude=47.7000000&height=11.000"
    }
  },
  "streamElements": [
    {
      "streamElement": {
```

```
        "frame": {
          "authority": "/geopose/1.0",
          "id": "RotateTranslate",
          "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[-0.90510, 0.
20057, -0.08112, 0.36605]"
        },
        "validTime": 1630560671474
      }
    },
    {
      "streamElement": {
        "frame": {
          "authority": "/geopose/1.0",
          "id": "RotateTranslate",
          "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[-0.90566, 0.
20166, -0.08106, 0.36406]"
        },
        "validTime": 1630560683820
      }
    },
    {
      "streamElement": {
        "frame": {
          "authority": "/geopose/1.0",
          "id": "RotateTranslate",
          "parameters": "translation=[0.0, 0.0, 0.0]&rotation=[-0.90622, 0.
20275, -0.08101, 0.36207]"
        },
        "validTime": 1630560696165
      }
    }
  ]
}
```

10

MEDIA TYPES FOR JSON ENCODING

10 MEDIA TYPES FOR JSON ENCODING

application/json: http://www.iana.org/assignments/media-types/application/json

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A.1. Introduction

GeoPose 1.0 specifies eight Standardization Targets using elements of the Logical Model. These elements are Structural Data Units (SDUs) and they have the stereotype “Structural Data Unit — SDU”. Each SDU is an element of the Logical Model that will be expressed in concrete data objects encoded using specific encoding or serialization technologies.

Although implementation of the Standardization Targets, expressed as SDUs, is independent of the Logical Model, GeoPose 1.0 also defines one of many possible implementations, single encoding in JavaScript Object Notation (JSON). The encodings of the eight targets are specified using Internet-Draft draft-handrews-json-schema-02. To keep the individual Standardization targets independent, there are some SDU requirements and corresponding conformance tests that appear in more than one requirement or conformance class. This structure is based on the judgement that it is easier to understand the independence of targets with complete definitions than would be the case if the definitional requirements of the SDUs were factored out and referenced indirectly by individual encodings.

A.2. Global conformance class

Conformance with the Global Requirements is required for all implementations.

CONFORMANCE CLASS A.1: GLOBAL SDU CONFORMANCE	
`/conf/global`	
Requirements class	Requirements class 1: `/req/global`
Conformance with global SDU requirements	
Conformance test	Conformance test A.1: `/req/global/target-independence`

CONFORMANCE CLASS A.1: GLOBAL SDU CONFORMANCE	
Conformance test	Conformance test A.2: `/req/global/logical-model`
Conformance test	Conformance test A.3: `/req/global/sdu`

CONFORMANCE TEST A.1: VERIFY INDIVIDUAL STANDARDIZATION TARGETS ARE INDEPENDENT	
`/req/global/target-independence`	
Requirement	Conformance test A.1: `/req/global/target-independence`
Included in	Conformance class A.1: `/conf/global`
Test purpose	Verify standardization targets are independent.
Test method	Inspection

CONFORMANCE TEST A.2: VERIFY IMPLEMENTATION CONFORMS TO THE LOGICAL MODEL	
`/req/global/logical-model`	
Requirement	Conformance test A.2: `/req/global/logical-model`
Included in	Conformance class A.1: `/conf/global`
Implementations of concrete data conforming to this standard SHALL conform to all dependent or inherited classes, attributes, and associations, multiplicities, and data types in the Logical Model.	
Test purpose	Verify implementation conforms to the logical model.
Test method	Inspection

CONFORMANCE TEST A.3: VERIFY SDU CONFORMS TO THE "STRUCTURAL DATA UNIT — SDU" STEREOTYPE	
`/req/global/sdu`	
Requirement	Conformance test A.3: `/req/global/sdu`

CONFORMANCE TEST A.3: VERIFY SDU CONFORMS TO THE "STRUCTURAL DATA UNIT — SDU" STEREOTYPE	
Included in	Conformance class A.1: `/conf/global`
Implementations using encoded SDUs SHALL conform to the logical description of the Logical Model elements with the "Structural Data Unit - SDU" stereotype.	
Test purpose	Verify SDU conforms to the "Structural Data Unit — SDU" stereotype.
Test method	Inspection

A.3. Common conformance classes

A.3.1. Tangent point specification conformance class

CONFORMANCE CLASS A.2: TANGENT POINT CONFORMANCE	
`/conf/tangent-point`	
Requirements class	Requirements class 2: `/req/tangent-point`
Conformance with tangent point requirements	
Conformance test	Conformance test A.4: `/conf/tangent-point/height`
Conformance test	Conformance test A.5: `/conf/tangent-point/latitude`
Conformance test	Conformance test A.6: `/conf/tangent-point/longitude`

CONFORMANCE TEST A.4: VERIFY TANGENT POINT HEIGHT VALUE MEETS SPECIFICATION	
`/conf/tangent-point/height`	
Requirement	Requirement 4: `/req/tangent-point/height`
Included in	Conformance class A.2: `/conf/tangent-point`
To confirm that a GeoPose `tangentPoint.h` attribute is expressed as a height in meters above the WGS-84 ellipsoid and represented as a signed real number.	

CONFORMANCE TEST A.4: VERIFY TANGENT POINT HEIGHT VALUE MEETS SPECIFICATION	
Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

CONFORMANCE TEST A.5: VERIFY TANGENT POINT LATITUDE VALUE MEETS SPECIFICATION	
`/conf/tangent-point/latitude`	
Requirement	Requirement 5: `/req/tangent-point/latitude`
Included in	Conformance class A.2: `/conf/tangent-point`
To confirm that a GeoPose `tangentPoint.latitude` attribute is expressed as an angle in decimal degrees.	
Test purpose	Verify that this requirement is satisfied
Test method	Inspection

CONFORMANCE TEST A.6: VERIFY TANGENT POINT LONGITUDE VALUE MEETS SPECIFICATION	
`/conf/tangent-point/longitude`	
Requirement	Requirement 6: `/req/tangent-point/longitude`
Included in	Conformance class A.2: `/conf/tangent-point`
To confirm that a GeoPose `tangentPoint.longitude` attribute is expressed as an angle in decimal degrees.	
Test purpose	Verify that this requirement is satisfied
Test method	Inspection

A.3.2. Frame specification conformance class

CONFORMANCE CLASS A.3: FRAME SPECIFICATION REQUIREMENTS

`/conf/frame-spec`

Requirements class	Requirements class 3: `/req/frame-spec`
Conformance with frame specification requirements	
Conformance test	Conformance test A.7: `/conf/frame-spec/authority`
Conformance test	Conformance test A.8: `/conf/frame-spec/id`
Conformance test	Conformance test A.9: `/conf/frame-spec/parameters`

CONFORMANCE TEST A.7: VERIFY FRAME SPECIFICATION AUTHORITY UNIQUELY SPECIFIES SOURCE OF REFERENCE FRAME SPECIFICATION

`/conf/frame-spec/authority`

Requirement	Requirement 7: `/req/frame-spec/authority`
Included in	Conformance class A.3: `/conf/frame-spec`
To confirm the correct properties of a Frame Specification Authority.	
Test purpose	To confirm that a FrameSpecification.authority attribute contains a string uniquely specifying a source of reference frame specifications.
Test method	Inspection

CONFORMANCE TEST A.8: FRAME SPECIFICATION ID UNIQUELY DEFINES FRAME WITHIN AUTHORITY

`/conf/frame-spec/id`

Requirement	Requirement 8: `/req/frame-spec/id`
Included in	Conformance class A.3: `/conf/frame-spec`
To confirm the correct properties of a Frame Specification ID.	
Test purpose	To confirm that a FrameSpecification.id attribute contains a string uniquely specifying the identity of a reference frame specification as defined by that authority.

CONFORMANCE TEST A.8: FRAME SPECIFICATION ID UNIQUELY DEFINES FRAME WITHIN AUTHORITY	
Test method	Inspection

CONFORMANCE TEST A.9: FRAME SPECIFICATION PARAMETER CONTAINS ALL PARAMETERS NEEDED	
`/conf/frame-spec/parameters`	
Requirement	Requirement 9: `/req/frame-spec/parameters`
Included in	Conformance class A.3: `/conf/frame-spec`
To confirm the correct properties of Frame Specification Parameters.	
Test purpose	To confirm that a FrameSpecification.parameters attribute contains contain all parameters needed for the corresponding authority and ID.
Test method	Inspection

A.3.3. Time specification conformance class

CONFORMANCE CLASS A.4: TIME SPECIFICATION REQUIREMENTS	
`/conf/time`	
Requirements class	Requirements class 4: `/req/time`
Conformance with GeoPose time-related requirements	
Conformance test	Conformance test A.10: `/conf/time/instant`
Conformance test	Conformance test A.11: `/conf/time/duration`

CONFORMANCE TEST A.10: VERIFY VALUES OF GEOPOSE_INSTANT	
`/conf/time/instant`	
Requirement	Requirement 10: `/req/time/instant`

CONFORMANCE TEST A.10: VERIFY VALUES OF GEOPOSE_INSTANT	
Included in	Conformance class A.4: `/conf/time`
Verify that a `GeoPose_Instant` object expresses Unix time in seconds multiplied by 1,000, with the unit of measure in milliseconds.	
Test purpose	To confirm the correct properties of a Geo Pose_Instant object.
Test method	Inspection

CONFORMANCE TEST A.11: VERIFY VALUES OF GEOPOSE_DURATION	
`/conf/time/duration`	
Requirement	Requirement 11: `/req/time/duration`
Included in	Conformance class A.4: `/conf/time`
Verify that a `GeoPose_Duration` object expresses time in seconds multiplied by 1,000, with the unit of measure in milliseconds.	
Test purpose	To confirm that a FrameSpecification.id attribute contains a string uniquely specifying the identity of a reference frame specification as defined by that authority.
Test method	Inspection

A.4. SDU conformance

A.4.1. General

There are some universal requirements on values that appear in a concrete implementation using a specific encoding technology. For example, angles may be constrained to fall within a range of values corresponding to a circle. Because these are independent of encoding technology, they are specified here at a logical level. Tests of an implementation at the SDU level generally only be done by inspection.

A.4.2. Basic-YPR SDU Conformance Class

CONFORMANCE CLASS A.5: BASIC-YPR LOGICAL MODEL SDU CONFORMANCE

`/conf/basic-ypr`

Requirements class	Requirements class 5: `/req/basic-ypr`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.2: `/conf/tangent-point`

Conformance with Basic-YPR logical model SDU

Conformance test	Conformance test A.12: `/conf/basic-ypr/position`
Conformance test	Conformance test A.13: `/conf/basic-ypr/angles`

CONFORMANCE TEST A.12: VERIFY EXPRESSION OF OUTER FRAME

`/conf/basic-ypr/position`

Requirement	Requirement 12: `/req/basic-ypr/position`
Included in	Conformance class A.5: `/conf/basic-ypr`

To confirm that an implementation of a Basic-YPR consists of an Outer Frame specified by an implicit WGS-84 CRS and an implicit EPSG 4461-CS (LTP-ENU) coordinate system and explicit parameters to define the tangent point.

Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

CONFORMANCE TEST A.13: VERIFY EXPRESSION OF INNER FRAME

`/conf/basic-ypr/angles`

Requirement	Requirement 13: `/req/basic-ypr/angles`
Included in	Conformance class A.5: `/conf/basic-ypr`

To confirm that the Inner Frame is expressed as a rotation-only transformation using Yaw, Pitch, and Roll angles. To confirm that GeoPose YPR angles are expressed as three consecutive rotations about the local axes Z, Y, and X, in that order, corresponding to the conventional Yaw, Pitch, and Roll angles and that the unit of measure is the degree.

CONFORMANCE TEST A.13: VERIFY EXPRESSION OF INNER FRAME	
Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

A.4.3. Basic-Quaternion SDU Conformance Class

CONFORMANCE CLASS A.6: BASIC-QUATERNION LOGICAL MODEL SDU CONFORMANCE	
`/conf/basic-quaternion`	
Requirements class	Requirements class 6: `/req/basic-quaternion`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.2: `/conf/tangent-point`
Conformance with Basic-Quaternion logical model SDU	
Conformance test	Conformance test A.14: `/conf/basic-quaternion/position`
Conformance test	Conformance test A.15: `/conf/basic-quaternion/quaternion`

CONFORMANCE TEST A.14: VERIFY EXPRESSION OF OUTER FRAME	
`/conf/basic-quaternion/position`	
Requirement	Requirement 14: `/req/basic-quaternion/position`
Included in	Conformance class A.6: `/conf/basic-quaternion`
To confirm that a Basic-Quaternion GeoPose contains an Outer Frame specified by an implicit WGS-84 CRS and an implicit EPSG 4461-CS (LTP-ENU) coordinate system and explicit parameters defining the tangent point.	
Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

CONFORMANCE TEST A.15: VERIFY EXPRESSION OF INNER FRAME	
`/conf/basic-quaternion/quaternion`	
Requirement	Requirement 15: `/req/basic-quaternion/quaternion`
Included in	Conformance class A.6: `/conf/basic-quaternion`
To confirm that a Basic-Quaternion GeoPose contains an Inner Frame is a rotation-only transformation using a unit quaternion, which is an instance of a GeoPose Logical Model quaternion data type value that is expressed as four real numbers, representing four quaternion components w, x, y, z, in that sequential order. The sum of the squares of the individual components is as close to 1.0 as the real number representation allows. The quaternion is applied to an initial reference frame oriented East-North-Up (ENU) coordinate system where the coordinate axes East, North, and Up correspond to the axes X, Y, Z.	
Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

A.4.4. Advanced SDU Conformance Class

CONFORMANCE CLASS A.7: BASIC-YPR LOGICAL MODEL SDU CONFORMANCE	
`/conf/advanced`	
Requirements class	Requirements class 7: `/req/advanced`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.3: `/conf/frame-spec`
Dependency	Conformance class A.4: `/conf/time`
To confirm that an implementation of the Advanced GeoPose conforms to the Logical Model.	
Conformance test	Conformance test A.16: `/conf/advanced/valid-time`
Conformance test	Conformance test A.17: `/conf/advanced/frame-spec`
Conformance test	Conformance test A.18: `/conf/advanced/quaternion`

CONFORMANCE TEST A.16: VERIFY EXPRESSION OF VALID TIME AS GEOPOSE_INSTANT

`/conf/advanced/valid-time`

Requirement	Requirement 16: `/req/advanced/valid-time`
Included in	Conformance class A.7: `/conf/advanced`
Dependency	Conformance test A.10: `/conf/time/instant`
To confirm the correct properties of a GeoPose_Instant.	
Test purpose	Confirm that the `Advanced.validTime` attribute is represented by a `GeoPose_Instant` object.
Test method	Inspection

CONFORMANCE TEST A.17: VERIFY EXPRESSION OF OUTER FRAME

`/conf/advanced/frame-spec`

Requirement	Requirement 17: `/req/advanced/frame-spec`
Included in	Conformance class A.7: `/conf/advanced`
To confirm that an implementation of an Advanced SDU contains an explicit frame specification in frameSpecification using the `ExplicitFrameSpec` object.	
Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

CONFORMANCE TEST A.18: VERIFY EXPRESSION OF INNER FRAME

`/conf/advanced/quaternion`

Requirement	Requirement 18: `/req/advanced/quaternion`
Included in	Conformance class A.7: `/conf/advanced`
To confirm that the unit quaternion consists of four representations of real number values and that the square root of the sum of the squares of those numbers is approximately 1.	

CONFORMANCE TEST A.18: VERIFY EXPRESSION OF INNER FRAME	
Test purpose	To confirm the correct properties of a quaternion.
Test method	Inspection

A.4.5. Graph SDU Conformance Class

CONFORMANCE CLASS A.8: GRAPH LOGICAL MODEL SDU CONFORMANCE	
`/conf/graph`	
Requirements class	Requirements class 8: `/req/graph`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.3: `/conf/frame-spec`
To confirm that an implementation of the GeoPose Graph conforms to the Logical Model.	
Conformance test	Conformance test A.19: `/conf/graph/valid-time`
Conformance test	Conformance test A.20: `/conf/graph/frame-list`
Conformance test	Conformance test A.21: `/conf/graph/transform-list`

CONFORMANCE TEST A.19: VERIFY EXPRESSION OF VALID TIME AS GEOPOSE_INSTANT	
`/conf/graph/valid-time`	
Requirement	Requirement 19: `/req/graph/valid-time`
Included in	Conformance class A.8: `/conf/graph`
Dependency	Conformance test A.10: `/conf/time/instant`
To confirm the correct properties of a GeoPose_Instant.	
Test purpose	Confirm that the `Graph.validTime` attribute is represented by a `GeoPose_Instant` object.
Test method	Inspection

CONFORMANCE TEST A.20: VERIFY LIST OF FRAME SPECIFICATIONS	
`/conf/graph/frame-list`	
Requirement	Requirement 20: `/req/graph/frame-list`
Included in	Conformance class A.8: `/conf/graph`
To confirm that an implementation of an Graph SDU contains a list of explicit frame specifications as `ExplicitFrameSpec` objects.	
Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

CONFORMANCE TEST A.21: VERIFY TRANSFORMS FOR FRAME SPECIFICATION LIST	
`/conf/graph/transform-list`	
Requirement	Requirement 21: `/req/graph/transform-list`
Included in	Conformance class A.8: `/conf/graph`
To confirm that each index value in a FrameListTransformPair is a distinct integer value between 0 and one less than the number of elements in the frameList property.	
Test purpose	To confirm that an implementation of Graph Index conforms to the Logical Model.
Test method	Inspection

A.4.6. Chain SDU Conformance Class

CONFORMANCE CLASS A.9: CHAIN LOGICAL MODEL SDU CONFORMANCE	
`/conf/chain`	
Requirements class	Requirements class 9: `/req/chain`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.3: `/conf/frame-spec`

CONFORMANCE CLASS A.9: CHAIN LOGICAL MODEL SDU CONFORMANCE	
Dependency	Conformance class A.4: `/conf/time`
To confirm that an implementation of the GeoPose Chain conforms to the Logical Model.	
Conformance test	Conformance test A.22: `/conf/chain/valid-time`
Conformance test	Conformance test A.23: `/conf/chain/initial-frame`
Conformance test	Conformance test A.24: `/conf/chain/frame-chain`

CONFORMANCE TEST A.22: VERIFY EXPRESSION OF VALID TIME AS GEOPOSE_INSTANT	
`/conf/chain/valid-time`	
Requirement	Requirement 22: `/req/chain/valid-time`
Included in	Conformance class A.9: `/conf/chain`
Dependency	Conformance test A.10: `/conf/time/instant`
To confirm the correct properties of a GeoPose_Instant.	
Test purpose	Confirm that the `Chain.validTime` attribute is represented by a `GeoPose_Instant` object.
Test method	Inspection

CONFORMANCE TEST A.23: VERIFY SPECIFICATION OF INITIAL FRAME	
`/conf/chain/initial-frame`	
Requirement	Requirement 23: `/req/chain/initial-frame`
Included in	Conformance class A.9: `/conf/chain`
To confirm that an implementation of an Chain SDU contains an initial frame under `Chain.outerFrame`.	
Test purpose	Verify that this requirement is satisfied.
Test method	Inspection

CONFORMANCE TEST A.24: VERIFY CHAIN OF FRAME SPECIFICATIONS	
`/conf/chain/frame-chain`	
Requirement	Requirement 24: `/req/chain/frame-chain`
Included in	Conformance class A.9: `/conf/chain`
To confirm that each index value in `Chain.frameChain` is a distinct integer value between 0 and one less than the number of elements in the frameChain property.	
Test purpose	To confirm that an implementation of Chain Index conforms to the Logical Model.
Test method	Inspection

A.4.7. Regular Series SDU Conformance Class

CONFORMANCE CLASS A.10: REGULAR_SERIES LOGICAL MODEL SDU CONFORMANCE	
`/conf/series-regular`	
Requirements class	Requirements class 10: `/req/series-regular`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.3: `/conf/frame-spec`
Dependency	Conformance class A.4: `/conf/time`
To confirm that components of a Regular Series conform to the Logical Model.	
Conformance test	Conformance test A.25: `/conf/series-regular/duration`
Conformance test	Conformance test A.26: `/conf/series-regular/outer-frame`
Conformance test	Conformance test A.27: `/conf/series-regular/inner-frame-series`
Conformance test	Conformance test A.28: `/conf/series-regular/header`
Conformance test	Conformance test A.29: `/conf/series-regular/trailer`

CONFORMANCE TEST A.25: VERIFY EXPRESSION OF DURATION AS GEOPOSE_DURATION

`/conf/series-regular/duration`

Requirement	Requirement 25: `/req/series-regular/duration`
Included in	Conformance class A.10: `/conf/series-regular`
Dependency	Conformance test A.11: `/conf/time/duration`
To confirm that the `Regular_Series.interPoseDuration` attribute is represented by an instance of the `GeoPose_Duration` object.	
Test purpose	To confirm the correct properties of a GeoPose Duration.

CONFORMANCE TEST A.26: VERIFY EXPRESSION OF OUTER FRAME

`/conf/series-regular/outer-frame`

Requirement	Requirement 26: `/req/series-regular/outer-frame`
Included in	Conformance class A.10: `/conf/series-regular`
Test purpose	The `Regular_Series.outerFrame` attribute shall represent the first frame in the series with the `ExplicitFrameSpec` object.

CONFORMANCE TEST A.27: VERIFY EXPRESSION OF INNER FRAMES

`/conf/series-regular/inner-frame-series`

Requirement	Requirement 27: `/req/series-regular/inner-frame-series`
Included in	Conformance class A.10: `/conf/series-regular`
Test purpose	The `Regular_Series.innerFrameSeries` attribute shall represent the succession of inner frames as an array of `ExplicitFrameSpec` objects.

CONFORMANCE TEST A.28: VERIFY EXPRESSION OF SERIES HEADER

`/conf/series-regular/header`

Requirement	Requirement 28: `/req/series-regular/header`

CONFORMANCE TEST A.28: VERIFY EXPRESSION OF SERIES HEADER	
Included in	Conformance class A.10: `/conf/series-regular`
Verify that the `Regular_Series.header` attribute is implemented as an instance of Series Header.	
Test purpose	To confirm that the implementation of Series Header conforms to the Logical Model.

CONFORMANCE TEST A.29: VERIFY EXPRESSION OF SERIES TRAILER	
`/conf/series-regular/trailer`	
Requirement	Requirement 29: `/req/series-regular/trailer`
Included in	Conformance class A.10: `/conf/series-regular`
Verify that the `Regular_Series.trailer` attribute is implemented as an instance of Series Trailer.	
Test purpose	To confirm that the implementation of SeriesTrailer conforms to the Logical Model.

A.4.8. Irregular Series SDU Conformance Class

CONFORMANCE CLASS A.11: IRREGULAR_ SERIES LOGICAL MODEL SDU CONFORMANCE	
`/conf/series-irregular`	
Requirements class	Requirements class 11: `/req/series-irregular`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.3: `/conf/frame-spec`
Dependency	Conformance class A.4: `/conf/time`
To confirm that components of an Irregular_Series conform to the Logical Model.	
Conformance test	Conformance test A.30: `/conf/series-irregular/inner-frame-and-time`
Conformance test	Conformance test A.31: `/conf/series-irregular/outer-frame`

CONFORMANCE CLASS A.11: IRREGULAR_ SERIES LOGICAL MODEL SDU CONFORMANCE

Conformance test	Conformance test A.32: `/conf/series-irregular/header`
Conformance test	Conformance test A.33: `/conf/series-irregular/trailer`

CONFORMANCE TEST A.30: VERIFY EXPRESSION OF INNER FRAMES AND TIME SERIES

`/conf/series-irregular/inner-frame-and-time`

Requirement	Requirement 30: `/req/series-irregular/inner-frame-and-time`
Included in	Conformance class A.11: `/conf/series-irregular`
Dependency	Conformance test A.10: `/conf/time/instant`

Confirm that the `Irregular_Series.innerFrameAndTime` attribute is implemented as an array of `FrameAndTimeElement` objects, each of which is a pair of `ExplicitFrameSpec` and `GeoPoseInstant` objects.

Test purpose	To confirm that the `innerFrameAndTime` attribute values are implemented in accordance with the Logical Model.

CONFORMANCE TEST A.31: VERIFY EXPRESSION OF OUTER FRAME

`/conf/series-irregular/outer-frame`

Requirement	Requirement 31: `/req/series-irregular/outer-frame`
Included in	Conformance class A.11: `/conf/series-irregular`
Test purpose	The `Irregular_Series.outerFrame` attribute shall represent the first frame in the series expressed by the `innerFrameAndTime` attribute.

CONFORMANCE TEST A.32: VERIFY EXPRESSION OF SERIES HEADER

`/conf/series-irregular/header`

Requirement	Requirement 32: `/req/series-irregular/header`
Included in	Conformance class A.11: `/conf/series-irregular`

CONFORMANCE TEST A.32: VERIFY EXPRESSION OF SERIES HEADER	
Verify that the `Irregular_Series.header` attribute is implemented as an instance of Series Header.	
Test purpose	To confirm that the implementation of Series Header conforms to the Logical Model.

CONFORMANCE TEST A.33: VERIFY EXPRESSION OF SERIES TRAILER	
`/conf/series-irregular/trailer`	
Requirement	Requirement 33: `/req/series-irregular/trailer`
Included in	Conformance class A.11: `/conf/series-irregular`
Verify that the `Irregular_Series.trailer` attribute is implemented as an instance of Series Trailer.	
Test purpose	To confirm that the implementation of SeriesTrailer conforms to the Logical Model.

A.4.9. Stream SDU Conformance Class

CONFORMANCE CLASS A.12: STREAMHEADER AND STREAMELEMENT LOGICAL MODEL SDUS CONFORMANCE	
`/conf/stream`	
Requirements class	Requirements class 12: `/req/stream`
Dependency	Conformance class A.1: `/conf/global`
Dependency	Conformance class A.3: `/conf/frame-spec`
To confirm that an implementation of the GeoPose Stream SDUs conforms to the Logical Model.	
Conformance test	Conformance test A.34: `/conf/stream/header-initial-frame`
Conformance test	Conformance test A.35: `/conf/stream/header-transition-model`
Conformance test	Conformance test A.36: `/conf/stream/element`

CONFORMANCE TEST A.34: VERIFY OUTER FRAME IN STREAMHEADER

`/conf/stream/header-initial-frame`

Requirement	Requirement 34: `/req/stream/header-initial-frame`
Included in	Conformance class A.12: `/conf/stream`
To confirm the correct specification of the `StreamHeader.outerFrame`.	
Test purpose	To confirm the initial frame is specified as an instance of the `ExplicitFrameSpec` object.
Test method	Inspection

CONFORMANCE TEST A.35: VERIFY TRANSITION MODEL IN STREAMHEADER

`/conf/stream/header-transition-model`

Requirement	Requirement 35: `/req/stream/header-transition-model`
Included in	Conformance class A.12: `/conf/stream`
To confirm the correct specification of the `StreamHeader.transitionModel`.	
Test purpose	To confirm the `transitionModel` is specified as an instance of the `TransitionModel` enumeration.
Test method	Inspection

CONFORMANCE TEST A.36: VERIFY STREAM ELEMENTS IN STREAMELEMENT

`/conf/stream/element`

Requirement	Requirement 36: `/req/stream/element`
Included in	Conformance class A.12: `/conf/stream`
Dependency	Conformance test A.10: `/conf/time/instant`

To confirm the correct specification of the `StreamElement`.

CONFORMANCE TEST A.36: VERIFY STREAM ELEMENTS IN STREAMELEMENT	
Test purpose	To confirm the StreamElement is implemented as an array of `FrameAndTimeElement` objects, each of which is a pair of `ExplicitFrameSpec` and `GeoPoseInstant` objects.
Test method	Inspection

A.5. Encoding conformance

A.5.1. JSON encoding

Each encoding technology has its own independent test suite. There is one conformance class per Standardization target per encoding technology. The GeoPose Standard 1.0 has one encoding technology: JSON.

A.5.2. Basic-YPR SDU JSON conformance class

The **Basic-YPR GeoPose** is the JSON encoding intended for widest use.

CONFORMANCE CLASS A.13: JSON ENCODING OF BASIC-YPR SDU	
`/conf/basic-ypr-encoding-json`	
Requirements class	Requirements class 13: `/req/basic-ypr-encoding-json`
Confirm that a JSON-encoded Basic-YPR GeoPose conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.37: `/conf/basic-ypr-encoding-json/definition`

CONFORMANCE TEST A.37: VERIFY CONFORMANCE VIA JSON SCHEMA	
`/conf/basic-ypr-encoding-json/definition`	
Requirement	Requirement 37: `/req/basic-ypr-encoding-json/definition`
Included in	Conformance class A.13: `/conf/basic-ypr-encoding-json`

CONFORMANCE TEST A.37: VERIFY CONFORMANCE VIA JSON SCHEMA	
To confirm that a Basic-YPR GeoPose in JSON validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the Basic-YPR JSON Schema 2019-9 definition (Figure 23).

A.5.3. Basic-Quaternion SDU JSON conformance class

The **Basic-Quaternion GeoPose** JSON encoding is intended for applications using quaternions. It comes in two sub-versions: normal and strict. The only difference is that a strict sub-version does not allow additional JSON members.

CONFORMANCE CLASS A.14: STRICT JSON ENCODING OF BASIC-QUATERNION SDU	
`/conf/basic-quaternion-encoding-json-strict`	
Requirements class	Requirements class 14: `/req/basic-quaternion-encoding-json-strict`
Conformance with the strict JSON encoding of Basic-Quaternion SDU	
Conformance test	Conformance test A.38: `/conf/basic-quaternion-encoding-json-strict/definition`

CONFORMANCE TEST A.38: VERIFY CONFORMANCE VIA JSON SCHEMA	
`/conf/basic-quaternion-encoding-json-strict/definition`	
Requirement	Requirement 38: `/req/basic-quaternion-encoding-json-strict/definition`
Included in	Conformance class A.14: `/conf/basic-quaternion-encoding-json-strict`
To confirm that a Basic-Quaternion GeoPose in strict JSON encoding validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the strict Basic-Quaternion JSON-Schema 2019-9 definition (Figure 24).

NOTE: The **Basic-Quaternion (Strict) GeoPose** JSON encoding does not allow additional JSON members.

CONFORMANCE CLASS A.15: PERMISSIVE JSON ENCODING OF BASIC-QUATERNION SDU	
`/conf/basic-quaternion-encoding-json`	
Requirements class	Requirements class 15: `/req/basic-quaternion-encoding-json`
Confirm that a JSON-encoded Basic-Quaternion GeoPose conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.39: `/conf/basic-quaternion-encoding-json/definition`

CONFORMANCE TEST A.39: VERIFY CONFORMANCE VIA JSON SCHEMA	
`/conf/basic-quaternion-encoding-json/definition`	
Requirement	Requirement 39: `/req/basic-quaternion-encoding-json/definition`
Included in	Conformance class A.15: `/conf/basic-quaternion-encoding-json`
To confirm that a Basic-Quaternion GeoPose in JSON validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the Basic-Quaternion JSON Schema 2019-9 definition (Figure 25).

A.5.4. Advanced SDU JSON conformance class

The **Advanced GeoPose** JSON encoding has an optional time stamp and a flexible Outer Frame specification.

CONFORMANCE CLASS A.16: JSON ENCODING OF ADVANCED SDU	
`/conf/advanced-encoding-json`	
Requirements class	Requirements class 16: `/req/advanced-encoding-json`

CONFORMANCE CLASS A.16: JSON ENCODING OF ADVANCED SDU	
Confirm that a JSON-encoded Advanced GeoPose conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.40: `/conf/advanced-encoding-json/definition`

CONFORMANCE TEST A.40: VERIFY CONFORMANCE VIA JSON SCHEMA	
`/conf/advanced-encoding-json/definition`	
Requirement	Requirement 40: `/req/advanced-encoding-json/definition`
Included in	Conformance class A.16: `/conf/advanced-encoding-json`
To confirm that a Advanced GeoPose in JSON validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the Advanced JSON-Schema 2019-9 definition (Figure 26).

A.5.5. Graph SDU JSON conformance class

The **GeoPose Graph** JSON encoding supports a directed graph structure.

CONFORMANCE CLASS A.17: JSON ENCODING OF GRAPH SDU	
`/conf/graph-encoding-json`	
Requirements class	Requirements class 17: `/req/graph-encoding-json`
Confirm that a JSON-encoded GeoPose Graph conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.41: `/conf/graph-encoding-json/definition`

CONFORMANCE TEST A.41: VERIFY CONFORMANCE VIA JSON SCHEMA
`/conf/graph-encoding-json/definition`

CONFORMANCE TEST A.41: VERIFY CONFORMANCE VIA JSON SCHEMA	
Requirement	Requirement 41: `/req/graph-encoding-json/definition`
Included in	Conformance class A.17: `/conf/graph-encoding-json`
To confirm that a GeoPose Graph in JSON validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the GeoPose Graph JSON-Schema 2019-9 definition (Figure 27).

A.5.6. Chain SDU JSON conformance class

The **GeoPose Chain** JSON encoding supports a linear sequence of frame transformations for modelling articulated structures.

CONFORMANCE CLASS A.18: JSON ENCODING OF CHAIN SDU	
`/conf/chain-encoding-json`	
Requirements class	Requirements class 18: `/req/chain-encoding-json`
Confirm that a JSON-encoded GeoPose Chain conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.42: `/conf/chain-encoding-json/definition`

CONFORMANCE TEST A.42: VERIFY CONFORMANCE VIA JSON SCHEMA	
`/conf/chain-encoding-json/definition`	
Requirement	Requirement 42: `/req/chain-encoding-json/definition`
Included in	Conformance class A.18: `/conf/chain-encoding-json`
To confirm that a GeoPose Chain in JSON validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the GeoPose Chain JSON-Schema 2019-9 definition (Figure 28).

A.5.7. Regular Series SDU JSON conformance class

The **GeoPose Regular Series** JSON encoding supports a time series of equally-spaced GeoPoses.

CONFORMANCE CLASS A.19: JSON ENCODING OF REGULAR SERIES SDU

`/conf/series-regular-encoding-json`

Requirements class	Requirements class 19: `/req/series-regular-encoding-json`
Confirm that a JSON-encoded GeoPose Regular Series conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.43: `/conf/series-regular-encoding-json/definition`

CONFORMANCE TEST A.43: VERIFY CONFORMANCE VIA JSON SCHEMA

`/conf/series-regular-encoding-json/definition`

Requirement	Requirement 43: `/req/series-regular-encoding-json/definition`
Included in	Conformance class A.19: `/conf/series-regular-encoding-json`
To confirm that a GeoPose Regular Series in JSON validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the GeoPose Regular Series JSON-Schema 2019-9 definition (Figure 29).

A.5.8. Irregular Series SDU JSON conformance class

The **GeoPose Irregular Series** JSON encoding has an optional time stamp and a flexible Outer Frame specification.

CONFORMANCE CLASS A.20: JSON ENCODING OF IRREGULAR SERIES SDU

`/conf/series-irregular-encoding-json`

CONFORMANCE CLASS A.20: JSON ENCODING OF IRREGULAR SERIES SDU	
Requirements class	Requirements class 20: `/req/series-irregular-encoding-json`
Confirm that a JSON-encoded GeoPose Irregular Series conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.44: `/conf/series-irregular-encoding-json/definition`

CONFORMANCE TEST A.44: VERIFY CONFORMANCE VIA JSON SCHEMA	
`/conf/series-irregular-encoding-json/definition`	
Requirement	Requirement 44: `/req/series-irregular-encoding-json/definition`
Included in	Conformance class A.20: `/conf/series-irregular-encoding-json`
To confirm that a GeoPose Irregular Series in JSON validates against the JSON schema.	
Test purpose	Verify that data validates against the corresponding JSON schema.
Test method	Validate the JSON data against the GeoPose Irregular Series JSON-Schema 2019-9 definition (Figure 30).

A.5.9. Stream SDU JSON conformance class

The **GeoPose Stream** JSON encoding has an optional time stamp and a flexible Outer Frame specification.

CONFORMANCE CLASS A.21: JSON ENCODING OF STREAM SDUS	
`/conf/stream-encoding-json`	
Requirements class	Requirements class 21: `/req/stream-encoding-json`
Confirm that a JSON-encoded GeoPose Stream conforms to the relevant elements of the Logical Model and a corresponding JSON-Schema document.	
Conformance test	Conformance test A.45: `/conf/stream-encoding-json/element`
Conformance test	Conformance test A.46: `/conf/stream-encoding-json/header`

CONFORMANCE CLASS A.21: JSON ENCODING OF STREAM SDUS	
Conformance test	Conformance test A.47: `/conf/stream-encoding-json/record`

CONFORMANCE TEST A.45: VERIFY STREAM ELEMENT CONFORMANCE TO JSON SCHEMA	
`/conf/stream-encoding-json/element`	
Requirement	Requirement 45: `/req/stream-encoding-json/element`
Included in	Conformance class A.21: `/conf/stream-encoding-json`
To confirm that a GeoPose Stream Element in JSON validates against the JSON schema.	
Test purpose	Verify JSON data conforms to the JSON schema.
Test method	Validate the JSON data against the GeoPose Stream Element JSON-Schema 2019-9 definition (Figure 32).

CONFORMANCE TEST A.46: VERIFY STREAM HEADER CONFORMANCE TO JSON SCHEMA	
`/conf/stream-encoding-json/header`	
Requirement	Requirement 46: `/req/stream-encoding-json/header`
Included in	Conformance class A.21: `/conf/stream-encoding-json`
To confirm that a GeoPose Stream Header in JSON validates against the JSON schema.	
Test purpose	Verify JSON data conforms to the JSON schema.
Test method	Validate the JSON data against the GeoPose Stream Header JSON-Schema 2019-9 definition (Figure 31).

CONFORMANCE TEST A.47: VERIFY STREAM RECORD CONFORMANCE TO JSON SCHEMA	
`/conf/stream-encoding-json/record`	
Requirement	Requirement 47: `/req/stream-encoding-json/record`
Included in	Conformance class A.21: `/conf/stream-encoding-json`

CONFORMANCE TEST A.47: VERIFY STREAM RECORD CONFORMANCE TO JSON SCHEMA	
To confirm that a GeoPose Stream Record in JSON validates against the JSON schema.	
Test purpose	Verify JSON data conforms to the JSON schema.
Test method	Validate the JSON data against the GeoPose Stream Record JSON-Schema 2019-9 definition (Figure 33).

B

ANNEX B (INFORMATIVE) GEOPOSE LOCAL FRAME OF REFERENCE SPECIFICATIONS

B

ANNEX B (INFORMATIVE) GEOPOSE LOCAL FRAME OF REFERENCE SPECIFICATIONS

This annex has example Frame Specifications for the LTP-ENU and LTP-NED Reference Frames.

B.1. Local Tangent Plane — East North Up (LTP-ENU)

```
BASEGEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4979]],
CONVERSION["To LTP-ENU",
    METHOD["Geographic/topocentric conversions",
        ID["EPSG",9837]],
    PARAMETER["Latitude of topocentric origin",<latitude>,
        ANGLEUNIT["degree",0.0174532925199433],
        ID["EPSG",8834]],
    PARAMETER["Longitude of topocentric origin",<longitude>,
        ANGLEUNIT["degree",0.0174532925199433],
        ID["EPSG",8835]],
    PARAMETER["Ellipsoidal height of topocentric origin",<height>,
        LENGTHUNIT["metre",1],
        ID["EPSG",8836]]],
CS[Cartesian,3],
    AXIS["topocentric East (U)",east,
        ORDER[1],
        LENGTHUNIT["metre",1]],
    AXIS["topocentric North (V)",north,
        ORDER[2],
        LENGTHUNIT["metre",1]],
    AXIS["topocentric height (W)",up,
        ORDER[3],
        LENGTHUNIT["metre",1]],
USAGE[
    SCOPE["unknown"],
    AREA["To be specified"],
    BBOX[-90,-180,90,180]],
```

```
]
```

Figure B.1 — LTP-ENU ISO 19162 WKT

B.2. Local Tangent Plane — North East Down (LTP-NED)

```
BASEGEOGCRS["WGS 84",
    DATUM["World Geodetic System 1984",
        ELLIPSOID["WGS 84",6378137,298.257223563,
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    ID["EPSG",4979]],
CONVERSION["To LTP-NED",
    METHOD["Geographic/topocentric conversions",
        ID["EPSG",9837]],
    PARAMETER["Latitude of topocentric origin",<latitude>,
        ANGLEUNIT["degree",0.0174532925199433],
        ID["EPSG",8834]],
    PARAMETER["Longitude of topocentric origin",<longitude>,
        ANGLEUNIT["degree",0.0174532925199433],
        ID["EPSG",8835]],
    PARAMETER["Ellipsoidal height of topocentric origin",<height>,
        LENGTHUNIT["metre",1],
        ID["EPSG",8836]]],
CS[Cartesian,3],
    AXIS["topocentric North (U)",north,
        ORDER[1],
        LENGTHUNIT["metre",1]],
    AXIS["topocentric East (V)",east,
        ORDER[2],
        LENGTHUNIT["metre",1]],
    AXIS["topocentric depth (W)",down,
        ORDER[3],
        LENGTHUNIT["metre",1]],
USAGE[
    SCOPE["unknown"],
    AREA["To be specified"],
    BBOX[-90,-180,90,180]],
]
```

Figure B.2 — LTP-NED ISO 19162 WKT

C

ANNEX C (INFORMATIVE) GEOPOSE USE AND INTERPRETATION OF UNIX TIME

C

ANNEX C (INFORMATIVE) GEOPOSE USE AND INTERPRETATION OF UNIX TIME

The GeoPose Standard has adopted a variation UNIX time as the method for denoting the location of Instants on a timeline. The reasons for this specific choice include the widespread availability of UNIX time in computer operating systems, the straightforward conversion to UTC at the level of precision required by the use cases considered in GeoPose 1.0: 1 millisecond.

Clearly, applications requiring higher precision and the recognition of non-Newtonian physical processes would require a more complex treatment of time. This has been left to possible future versions of the GeoPose Standard.

C.1. Intended Precision

The intended precision of UNIX time in GeoPose 1.0 is 1 millisecond. Representations and encodings are based on the use of integer numbers of milliseconds.

C.2. Scaling

Time values are represented and encoded as integer values in GeoPose 1.0. Note that a computing environment must support 64-bit signed integers in order to accommodate the range of possible values.

C.3. Non-negative Time Positions

Times at or after the UNIX epoch of 1 January 1970 are represented as though clocks ticked forward with the same duration of a second as at the epoch. Conversion to time reference systems and calendars requires the consideration of the generally decreasing rate of rotation

of the earth with time increasing into the future. UTC, for example, makes use of leap seconds applied as needed either at 31 December or 30 June.

C.4. Negative Time Positions

Times before the UNIX epoch of 1 January 1970 are represented as though clocks ticked backward with the same duration of a second as at the epoch. Conversion to time reference systems and calendars requires the consideration of the generally increasing rate of rotation of the earth with time decreasing into the past. The rate is about 0.015 millisecond/year. The accumulated time error is about 0.6 second/year in the recent past.

C.5. Positive Time Positions before 1 January 1972 UTC

International timekeeping switched from an astronomical basis to a reference based on atomic processes in 1967. The details were in flux at the UNIX time epoch of 1 January 1970 until 1972, when the current system relating atomic time and UTC were adopted.

D

ANNEX D (INFORMATIVE) GLOSSARY

ANNEX D (INFORMATIVE) GLOSSARY

D.1. acceleration

time rate of change of *velocity* (Annex D.32)

D.2. accelerometer

sensor that can measure *acceleration* (Annex D.1)

Note 1 to entry: Low cost, accurate sensors for measuring 3 mutually perpendicular components of acceleration are widely deployed in vehicles, communications devices, and other connected devices.

D.3. angular acceleration

time rate of change of rotational *velocity* (Annex D.32)

D.4. application domain

context within which some technology or device is usefully applied

D.5. associated reference frame

pose frame ADMITTED

Euclidean *reference frame* (Clause 4.2.5) that is defined by the location and orientation of a *pose* (Clause 4.2.3)

Note 1 to entry: A pose defines the origin of its associated reference frame, and its *orientation* (Clause 4.2.2) defines the orientation of its associated reference frame.

Note 2 to entry: Associated reference frames are useful in many simulation and graphics applications where poses are most naturally defined in terms of another (parent) object's pose.

D.6. attribute

property (Annex D.27) associated with an object

Note 1 to entry: In object modelling, an attribute is the same as a property or data member.

D.7. barometric pressure

ambient pressure of the atmosphere at a location

Note 1 to entry: Low cost, accurate sensors for barometric pressure are widely deployed in connected devices.

Note 2 to entry: Sensing of changes in barometric pressure over time periods of minutes or less is enables estimation of vertical relative position.

D.8. Bluetooth indoor positioning service

indoor positioning service based on Bluetooth signal strength and/or triangulation

Note 1 to entry: A Bluetooth indoor positioning service allows precise determination of location and orientation inside smaller spaces.

Note 2 to entry: The location of a Bluetooth transceiver may be specified with respect to a geographic coordinate system and it may be possible to compute a GeoPose from interactions with multiple Bluetooth transceivers or other sensors.

D.9. 3-D cartesian coordinate system

cartesian coordinate system ADMITTED

system of geometrical reference using three mutually perpendicular axes where a point location is described by three numbers giving the perpendicular distance to each of the axes, all in the same numerical scale

D.10. class

template for the data structure and methods for operating on those data structures for objects sharing common characteristics

D.11. compass

sensor for measuring the relative orientation of a device to an ambient magnetic field

Note 1 to entry: Accurate and low-cost compasses are widely deployed in connected devices.

D.12. coordinate reference system

coordinate system referenced to a *datum* (Annex D.14)

D.13. data type

representational form for a concrete data element such as a number, character, or color

D.14.**datum**

reference point, line or surface used to establish measurements of position

Note 1 to entry: A datum is typically specified as a set of parameters that define the position of the origin, the scale, and the orientation of a coordinate system.

D.15.**geodetic datum**

datum (Annex D.14) that defines the measurement of horizontal position (latitude and longitude) and/or vertical position (height)

D.16.**ellipsoid**

mathematical surface that may be used as a *datum* (Annex D.14) in defining a geographic coordinate system

Note 1 to entry: An ellipsoid is usually established by fitting the parameters of the ellipsoid to measurements of a gravitational equipotential surface (*geoid* (Annex D.21)) that approximates mean sea level.

D.17.**east-north-up local tangent plane coordinate system**

ENU coordinate system ADMITTED

Euclidean 3-D coordinate system aligned with the Z-axis increasing upward, the X-axis aligned toward the direction east, and the Y-axis aligned toward north

Note 1 to entry: An ENU coordinate system is not defined at the poles because there is no inherent orientation.

D.18.**Euler angles**

description of the orientation of one Euclidean *reference frame* (Clause 4.2.5) to another by specifying the rotations about each of the three axes respectively to bring one in alignment with the other

D.19.**geographic coordinates**

3-dimensional reference system based on a reference *ellipsoid* (Annex D.16)

Note 1 to entry: Two of the coordinates are angles with respect to the axis of the ellipsoid and to a plane containing the axis of the ellipsoid and a specified point (principle point) on the ellipsoid surface. The third coordinate is a linear measure of height above the ellipsoidal surface.

D.20.**geographic position**

point defined in geographic coordinates

D.21.**geoid**

approximation of surface of equal gravitational force, usually attempting to match average sea-level

Note 1 to entry: A geoid is defined by measurements and is always inexact. The *ellipsoid* (Annex D.16) used in *geographic coordinate systems* (Annex D.19) is usually a mathematical approximation to a specific geoid.

D.22.**gyro**

sensor that measures the rate of rotation

Note 1 to entry: Low-cost, accurate Gyros are widely deployed in connected devices.

D.23.**kinematics**

properties of location, velocity, and acceleration of a body without regard to any forces acting on the body

D.24.**local tangent plane coordinate system**

LTP coordinate system ADMITTED

right-hand Euclidean coordinate system with a vertical (Z) axis extending from an origin at a point defined by geographic coordinates with respect to an *ellipsoid* (Annex D.16)

D.25.**local tangent plane east-north-up coordinate system**

local tangent plane east-north-up frame ADMITTED

LTP-ENU coordinate system ADMITTED

local tangent plane coordinate system (Annex D.24) specialized to an east-north-up system, where the X axis is aligned toward east and the Y axis toward north.

Note 1 to entry: While a *LTP coordinate system* (Annex D.24) can be established at any location, an ENU cannot be defined at the poles because it cannot be oriented.

D.26.**position**

location of a point with respect to the origin of a specific *reference frame* (Clause 4.2.5)

D.27.**property**

attribute (Annex D.6) associated with an object

Note 1 to entry: In object modelling, it is the same as an *attribute* (Annex D.6) or data member.

D.28.**quaternions**

extension of complex numbers

Note 1 to entry: Quaternions provide convenient properties for computing with rotations, in particular smooth interpolation and avoidance of "gimbal lock" possible with Euler Angles.

D.29.**rotation**

angular relationship between a reference frame's axes and a direction in that *reference frame* (Clause 4.2.5)

Note 1 to entry: *Euler angles* (Annex D.18), rotation matrices, and *quaternions* (Annex D.28) are three ways to specify a rotation.

D.30.**digital sensor**

sensor ADMITTED

device that converts environmental properties into data suitable for computation

D.31.**topographic surface**

interface between the liquid or solid surface of a planet and its atmosphere or surrounding empty space

Note 1 to entry: The topographic surface is always approximate. It may be measured with reference to a gravitational equipotential surface (such as a *geoid* (Annex D.21)) or a mathematical reference surface (such as an *ellipsoid* (Annex D.16)).

D.32.**velocity**

time rate of change of *position* (Annex D.26)

D.33.**vertical datum**

reference level from which elevation or altitude can be measured

Note 1 to entry: The *topographic surface* (Annex D.31), a *geoid* (Annex D.21), a level of constant *barometric pressure* (Annex D.7), or an *ellipsoid* (Annex D.16) are examples.

E

ANNEX E (INFORMATIVE) REVISION HISTORY

ANNEX E (INFORMATIVE) REVISION HISTORY

Table E.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2020-11-21	0.1.0	Steve Smyth	all	initial integrated version
2021-08-25	0.5.0	Steve Smyth	all	assigned doc number 21-056
2021-08-26	0.5.0	Steve Smyth	all	doc number 21-056r1
2021-09-08	0.6.0	Steve Smyth	all	doc number 21-056r2
2021-09-26	0.6.1	Steve Smyth	fix links and minor edits to all — does not include Unix Time in seconds	doc number 21-056r3
2021-09-28	0.6.2	Steve Smyth	fix section heading 7.3 (YPR)	2022-01-07
0.6.2	Steve Smyth	public review comment resolution	2022-01-27	0.7.0
Steve Smyth	late arriving public review comment resolution	2022-02-07	0.7.3	Steve Smyth
late arriving public review comment resolution	2022-02-07	0.9.0	Steve Smyth	final bit addressing late arriving public review comment resolution — all issues labelled “Public Review Comment” closed after this edit

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Leonard Daly, Scott Serich: OGC 20-087, *Interoperable Simulation and Gaming Sprint Engineering Report*. Open Geospatial Consortium (2021). https://docs.ogc.org/per/20-087.html.

2. Jérôme Jacovella-St-Louis: OGC 18-025, *OGC Testbed-14: CityGML and AR Engineering Report*. Open Geospatial Consortium (2019). https://docs.ogc.org/per/18-025.html.

3. CEN. ENV 1613:1994 Exchange of Laboratory Information.

4. ISO: ISO 8601-1:2019, *Date and time — Representations for information interchange — Part 1: Basic rules*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/70907.html.

5. ISO: ISO 8601-2:2019, *Date and time — Representations for information interchange — Part 2: Extensions*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/70908.html.

6. ISO: ISO 13972:2022, *Health informatics — Clinical information models — Characteristics, structures and requirements*. International Organization for Standardization, Geneva (2022). https://www.iso.org/standard/79498.html.

7. ISO: ISO 19111:2019, *Geographic information — Referencing by coordinates*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/74039.html.

8. ISO: ISO 19162:2019, *Geographic information — Well-known text representation of coordinate reference systems*. International Organization for Standardization, Geneva (2019). https://www.iso.org/standard/76496.html.

9. Chris Little, Simon Cox (eds.): W3C CR-owl-time-20200326, *Time Ontology in OWL*. (2020). https://www.w3.org/TR/2020/CR-owl-time-20200326/.

10. Booch, G., Rumbaugh, J., Jacobson, I.: Unified Modeling Language User Guide. Addison-Wesley (1997)

11. EPSG https://epsg.org

12. NASA SPICE https://naif.jpl.nasa.gov/naif/ Navigation and ancillary information facility

13. YPR https://www.gwg.nga.mil/misb//docs/standards/ST0601.8.pdf

14. OGC: OGC Testbed 12 Annex B: Architecture. (2015).

15. PHILIP J. SCHNEIDER, DAVID H. EBERLY, Geometric Tools for Computer Graphics, Morgan Kaufmann, 2003.

16. ISG Year 2 ER (https://github.com/opengeospatial/OGC-ISG-Sprint-Sep-2020/tree/master/Sprint%20Report)

17. GeoVolumes charter (https://portal.ogc.org/files/97171)

18. Reference frame https://en.wikipedia.org/wiki/Frame_of_reference

19. Pose https://en.wikipedia.org/wiki/Pose_(computer_vision)