OGC® DOCUMENT: 22-022R1

External identifier of this OGC® document: http://www.opengis.net/doc/is/sensorthings-staplus/1.0

OGC SENSORTHINGS API EXTENSION: STAPLUS 1.0

STANDARD
Implementation

APPROVED

Version: 1.0
Submission Date: 2023-02-14
Approval Date: 2023-08-29
Publication Date: 2023-09-23
Editor: Andreas Matheus

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I ABSTRACT

The OGC SensorThings API Extension: STAplus 1.0 Standard specifies a backwards-compatible extension to the OGC Standard SensorThings API Part 1: Sensing and Sensing 1.1 data model.

The motivation for specifying this STAplus extension is based on requirements from the Citizen Science community.

The dominant use for the OGC SensorThings API data model and API can be coined with the use case “single authority provides sensor readings to consumers.” However, in Citizen Science there are many contributors (citizens) who – together – create the big picture with their observations.

The STAplus extension is designed to support a model in which observations are owned by different users. This results in requirements for the ownership concept. In addition to the ownership, users may express a license for ensuring proper re-use of their observations. The STAplus extension also supports expressing explicit relations between observations as well as between observations and external resources. Relations can enrich observations to enable future extensions supporting Linked Data, RDF and SPARQL. Observation group(s) allow the grouping of observations that belong together.

The STAplus extension is believed to be an important contribution towards the realization of the FAIR principles as STAplus strengthens the “I” (Interoperability) through a common data model and API as well as the “R” (Re-usability) by allowing expressing standards-based queries that may consider licensing conditions which is relevant for reuse of other users’ observations.

The STAplus Data Model and Business Logic also enriches existing deployments as the extension can be seamlessly added and thereby offers new capabilities to create and manage the “big picture” with multi-user capabilities.

The key work for crafting this OGC Standard was undertaken in the *Co-designed Citizen Observatories Services for the EOS-Cloud* (Cos4Cloud) project, which received funding from the European Union’s Horizon 2020 research and innovation program and the *Enhancing Citizen Observatories for healthy, sustainable, resilient and inclusive cities* (CitiObs) project, which received funding from the European Union’s Horizon Europe research and innovation program. Testing of this extension was done with data from the Framework biodiversity project, which received funding from the European Union’s Horizon 2020 research and innovation program.

II KEYWORDS

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

OGC Standard, API, SensorThings, STAplus

III PREFACE

STAplus — OGC SensorThings API Extension: STAplus 1.0 — defines a SensorThings data model extension to improve FAIR principles when exchanging sensor data including licensing and ownership information.

The STAplus extension is fully backwards compatible to the existing OGC SensorThings API Part 1: Sensing as well as Sensing Version 1.1 and thereby offers existing deployments an easy upgrade to STAplus.

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

A STAplus service implementation that supports the Create, Update, or Delete of entities should also implement authentication and a "fit-for-purpose" business logic that enforces the required access conditions to ensure ownership of all entities, including SensorThings core entities.

The Business Logic makes the full power of STAplus useful! For example, sealed (closed) ObservationGroups — aka a group of observations that may include relations — could be used for interdisciplinary research by simply exchanging the group's URL. But, it is essential to trust such a group which requires a verifiable group author and the business logic to support integrity of such a closed group.

As each business logic flow may be different from service to service deployment, the operator of the service should describe the business logic and make it available to developers that intend to use the STAplus service. The specification of a conformance class `Business Logic`, reflecting the semantics of the logic in a standardized and machine readable fashion, is out of scope for this Standard but could be defined in an extension to STAplus.

Without an appropriate business logic, enforcing ownership and ensuring integrity in a multi-user scenario, it is possible that "junk" or "spam" is associated to a party without their knowledge. In such a multi-user CRUD access scenario, it would also be possible to "steal" an entire `Datastream` entity by simply updating the associated `Party` entity. Further, observations could be modified or even deleted without the owning user's approval. While not specific to STAplus, these Business Logic considerations and concerns should be identical for any deployed SensorThings API service.

The STAplus data model does not support the recording of provenance. If the business logic allows the update or even deletion of entities, there is no history of who did what and when. The data served from a service implementation is a snapshot in time which may cause pagination to produce non deterministic results.

For implementations of the `Create`, `Update`, and `Delete` conformance class it is paramount to check all user uploads for malicious content. Typical SQL injection checks are mandatory but also JavaScript code injection must be tackled. For example, it is likely that the value of an observation will be displayed on some HTML page. To prevent malicious code or virus injection, similar to cross-site-scripting attacks, pattern checking on any uploaded data should be accomplished.

The STAplus data model does **not** support the storage of personal data with a `Party` entity. Neither offers the `Party` entity the usual `properties` property to ensure personal data is not stored there. As a result, any requirement for accessing and sharing of personal data must take place with another system. Such system is likely connected to the same authentication as the STAplus service to ensure alignment between the `authId` within STAplus and the management of personal data stored at *the other system*. As personal data gets associated with STAplus user identifiers, it is recommended to operate *the other system* controlling the exchange of personal information. Such a system could be designed on an opt-in basis: A user is asking *the other system* for personal data to be used in a particular context of another user via the identifier. Once the user has approved the use of the personal data for the given context, the data received from STAplus can be linked with the personal data from *the other system* based on the user's identifier (`authId`). Even though the STAplus data model does not support the direct storage of personal data, it could be possible that in a wider eco-system personal data can be linked with user's identifier from STAplus.

The use of a unique `authId` is mandatory to ensure integrity and confidentiality of data when STAplus is used as a multi-user service. In case that multiple authentication systems are to be supported, the resulting `authId` has to be unique for the service. One possibility would be to require that acceptable authentication systems support the Decentralized Identifiers (DIDs) v1.0. The use of World Wide Web Consortium (W3C) Decentralized Identifiers (DIDs) helps by the DID structure that identifiers of each authentication system become globally unique.

The `authId` or the `displayName` can be used in `$filter` or `$select` to craft STAplus queries on entities that belong to a particular user. However, it is important to recognize that the `authId` for a user will not change but the `displayName` may.

Even though not a direct feature of STAplus, storage of location and time is possible as inherited from STA. An implementation's business logic may obfuscate the location of a sensor or observations based on internal logic. It is also possible that the user's application that uploads the time/location values obfuscates values according to user's discretion. This would ensure disconnect between location of observations and tracking location of people.

The exposure of time/location may become sensitive information in certain cases. The business logic should implement certain conditions that at least ensure compliance with the law.

The upload of binary data or graphical language must be sanitized in compliance with the law. In particular, ethical aspects must be considered to ensure that observation values are not misused i.e., to result into personal harassment.

V SUBMITTING ORGANIZATIONS

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

- Secure Dimensions GmbH
- Centre de Recerca Ecològica i Aplicacions Forestals (CREAF)
- Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.

VI SUBMITTERS

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

NAME	REPRESENTING	OGC MEMBER
Andreas Matheus	Secure Dimensions GmbH	Yes
Joan Maso	Centre de Recerca Ecològica i Aplicacions Forestals (CREAF)	Yes
Hylke van der Schaaf	Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.	Yes

1 SCOPE

1 SCOPE

While the OGC SensorThings API provides an open Standard-based and geospatial-enabled framework to interconnect the Internet of Things, OGC STAplus extends the scope of SensorThings API to use cases where sensors are deployed by different actors structured in one or more campaigns. The STAplus Standard also allows for relating observations to other internal or external observations and group observations. The open STAplus Standard is backwards-compatible with SensorThings API and uses the same web API and MQTT mechanisms. The grouping and relations can be used for linking sensor data beyond the original link to semantic definitions in ObservedProperty and unitOfMeasurement, and includes terms of use and licenses. Both additions improve the reusability aspect of the FAIR principles. This Standard also emphasizes on how to use an implementation in multi-user environments.

2

CONFORMANCE

2 CONFORMANCE

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

2.1. Introduction

The STAplus 1.0 standard defines an extension to the OGC Standards OGC SensorThings API Part 1: Sensing (SensorThings API v1.0) and OGC SensorThings API Part 1: Sensing Version 1.1 (SensorThings API v1.1) by adding additional entity types to the data model. Access to the STAplus entities via HTTP is defined in the Read, Create, Update, and Delete Requirements Classes. The use of MQTT for STAplus entities is defined in the MQTT Requirements Class. A default default-CRS is defined via the Default-CRS Requirements Class and additional geometry encodings are defined in the Geometry FG and Geometry WKT Requirements Class. The Authentication Requirements Class supports the unique identification of acting users. The Business Logic Requirements Class supports the textual description of the implemented business logic. An example of such a Business Logic is described in the OGC Best Practice for using SensorThings API with Citizen Science.

2.2. STAplus 1.0 Conformance Classes

This OGC Standard defines one mandatory and several optional conformance classes.

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 and Procedures and the OGC Compliance Testing web site.

In order to conform to this OGC Standard, a software implementation shall choose to implement:

- The one mandatory conformance class `Core`;
- Any one of the optional conformance classes; and
- Pass all applicable tests as defined in Annex A.

This is the list of STAplus Conformance Classes.

- **Core** (mandatory): This Conformance Class incorporates the capabilities to support the Read of STAplus entities via HTTP. It also includes the Default-CRS Requirements Class to ensure interoperability with geometry encodings in the `feature` and `location` properties.
- **Create** (optional): This Conformance Class supports to create STAplus entities via HTTP.
- **Update** (optional): This Conformance Class supports to update STAplus entities via HTTP.
- **Delete** (optional): This Conformance Class supports to delete STAplus entities via HTTP.
- **Authentication** (optional): This Conformance Class supports user authentication and their unique identification.
- **Geometry-FG** (optional): This Conformance Class supports the alternative geometry encoding for the `feature` and `location` properties based on Features and their Geometries.

NOTE: This Conformance Class remains informative until OGC Features and Geometries JSON - Part 1: Core is an adopted OGC Standard.

- **Geometry-WKT** (optional): This Conformance Class supports the alternative geometry encoding for the `feature` and `location` properties based on Well Known Text (WKT) as defined in Geographic information - Simple features access - Part 1: Common architecture.
- **MQTT Subscribe STA v1.0** (optional): This Conformance Class supports a client to receive STAplus entity changes via MQTT as defined in OGC SensorThings API Part 1: Sensing.
- **MQTT Subscribe STA v1.1** (optional): This Conformance Class supports a client to receive STAplus entity changes via MQTT as defined in OGC SensorThings API Part 1: Sensing Version 1.1.
- **Business Logic** (optional): This Conformance Class supports that the implementation's business logic is described in English text to help developers understand the details of CRUD access.

2.2.1. Conformance Class Core

The `Core` Conformance Class is defined as follows:

CONFORMANCE CLASS 1: CORE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
REQUIREMENTS CLASS	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information` Requirements class 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/party` Requirements class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/license`

CONFORMANCE CLASS 1: CORE	
	Requirements class 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/group` Requirements class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/relation` Requirements class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/campaign` Requirements class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/read` Requirements class 15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/default-crs`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/common-control-information` Conformance test A.2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/entities` Conformance test A.3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/read` Conformance test A.4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/crs-definition` Conformance test A.5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/axis-order` Conformance test A.6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/media-type` Conformance test A.7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/processing`

2.2.2. Conformance Class Create

The `Create` Conformance Class is defined as follows:

CONFORMANCE CLASS 2: CREATE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/create`
REQUIREMENTS CLASS	Requirements class 8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/create`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation

CONFORMANCE CLASS 2: CREATE	
CONFORMANCE TEST	Conformance test A.8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/create/http`

2.2.3. Conformance Class Update

The `Update` Conformance Class is defined as follows:

CONFORMANCE CLASS 3: UPDATE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update`
REQUIREMENTS CLASS	Requirements class 9: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/update`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.9: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update/put` Conformance test A.10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update/patch`

2.2.4. Conformance Class Delete

The `Delete` Conformance Class is defined as follows:

CONFORMANCE CLASS 4: DELETE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/delete`
REQUIREMENTS CLASS	Requirements class 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/delete`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation
CONFORMANCE TEST	Conformance test A.11: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/delete/entity`

2.2.5. Conformance Class Authentication

The `Authentication` Conformance Class is defined as follows:

CONFORMANCE CLASS 5: AUTHENTICATION	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`
REQUIREMENTS CLASS	Requirements class 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/authentication`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.12: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id` Conformance test A.13: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-create` Conformance test A.14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-read` Conformance test A.15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-update` Conformance test A.16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-delete`

2.2.6. Conformance Class Geometry FG

NOTE: This Conformance Class remains informative until OGC Features and Geometries JSON - Part 1: Core is an adopted OGC Standard.

The `Geometry FG` Conformance Class is defined as follows:

CONFORMANCE CLASS 6: GEOMETRY FG	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`
REQUIREMENTS CLASS	Requirements class 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`

CONFORMANCE CLASS 6: GEOMETRY FG	
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.19: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg//media-type` Conformance test A.20: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/default-crs` Conformance test A.21: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/supported-crs` Conformance test A.22: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/crs-error` Conformance test A.23: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/processing` Conformance test A.24: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/out`

2.2.7. Conformance Class Geometry WKT

The `Geometry WKT` Conformance Class is defined as follows:

CONFORMANCE CLASS 7: GEOMETRY WKT	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
REQUIREMENTS CLASS	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.25: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/media-type` Conformance test A.26: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/crs-definition` Conformance test A.27: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/default-crs` Conformance test A.28: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/supported-crs` Conformance test A.29: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/crs-error` Conformance test A.30: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/value` Conformance test A.31: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/processing`

CONFORMANCE CLASS 7: GEOMETRY WKT	
	Conformance test A.32: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/out`

2.2.8. Conformance Class MQTT Subscribe STA v1.0

The `MQTT Subscribe` STA v1.0 Conformance Class is defined as follows:

CONFORMANCE CLASS 8: MQTT SUBSCRIBE STA V1.0	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-10`
REQUIREMENTS CLASS	Requirements class 11: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/mqtt-subscribe-sta-10`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.33: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-10/definition` Conformance test 8-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-10`

2.2.9. Conformance Class MQTT Subscribe STA v1.1

The `MQTT Subscribe` STA v1.1 Conformance Class is defined as follows:

CONFORMANCE CLASS 9: MQTT SUBSCRIBE STA V1.1	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-11`
REQUIREMENTS CLASS	Requirements class 12: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/mqtt-subscribe-sta-11`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.34: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-11/definition`

CONFORMANCE CLASS 9: MQTT SUBSCRIBE STA V1.1	
	Conformance test 9-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-11`

2.2.10. Conformance Class Business Logic

The `Business Logic` Conformance Class is defined as follows:

CONFORMANCE CLASS 10: BUSINESS LOGIC	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic`
REQUIREMENTS CLASS	Requirements class 13: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/business-logic`
PREREQUISITE	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic/definition` Conformance test A.18: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic/location`

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.

OGC, *OGC SensorThings API Part 1: Sensing*, OGC (2016), https://docs.ogc.org/is/15-078r6/15-078r6.html

OGC, *OGC SensorThings API Part 1: Sensing Version 1.1*, OGC (2020), https://docs.ogc.org/is/18-088/18-088.html

IETF, *The GeoJSON Format*, (2016), https://www.rfc-editor.org/rfc/rfc7946.html

ISO, *Geographic information — Simple features access — Part 1: Common architecture*, 2004, https://portal.opengeospatial.org/files/?artifact_id=25355

4

TERMS, DEFINITIONS AND ABBREVIATED TERMS

4 TERMS, DEFINITIONS AND ABBREVIATED TERMS

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. Terms and definitions

4.1.1. **Campaign**

An event or an activity that results in or is linked to a collection of `MultiDatastream`, `Datastream`, and/or `ObservationGroup`. A campaign is commonly limited in space and time and can be managed by one party.

4.1.2. **License**

An agreement between a party and other parties to gain access and use primarily observations but also groups of observations and campaigns. The agreement can include, among others, clauses about how to share data with others not involved in the licensing agreement, the capability to execute commercial activities, the authorization to share modified versions of the data, and the need for attributing the original authors. Associating a license with a `MultiDatastream`, `Datastream` expresses re-use conditions for all associated observations. Associating a license to an `ObservationGroup` or a `Campaign` expresses their re-use conditions.

4.1.3. ObservationGroup

A collection of observations that are grouped together for a particular purpose.

4.1.4. Ownership

The right of possession over entities such as things, campaigns, observation groups, and other entities related to them. Ownership is required to properly manage create, update, and delete operations in a multi-user environment.

4.1.5. Party

A user, interacting with the service facilitating the creation of observations and managing owned resources. A party can own a thing, decide on the license of a datastream of observations, manage a campaign, be responsible of an observation group, etc.

4.1.6. Relation

A relationship established between two observations or an observation and an external resource for a particular reason.

4.2. Abbreviated terms

API	Application Programming Interface
CC	Creative Commons
CRUD	Create Read Update Delete
DID	Decentralized Identifier
GDPR	European General Data Protection Regulation

MQTT	Message Queuing Telemetry Transport
STA	Sensor Things API
STAplus	Sensor Things API PLUS extension

5

CONVENTIONS

5 CONVENTIONS

This Clause provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this standard are denoted by the URI

`http://www.opengis.net/spec/sensorthings-staplus/1.0`

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

6

INTRODUCTION

6 INTRODUCTION

The STAplus 1.0 Standard defines an extension to the OGC SensorThings API (STA). This STAplus extension originally started with motivations and requirements from the Citizen Science community. However, the STAplus extension has wider applicability than just Citizen Science.

STAplus is a 100% backwards compatible extension to the STA data model of version 1.0 and 1.1 and as such can be added to existing STA deployments.

In addition to the data model extension, this Standard defines different concepts that support operating an implementation in a multi-user CRUD deployment.

6.1. Concept of Ownership

In Citizen Science, users participate in campaigns or projects, offered by different Citizen Science portals. These portals, typically operated by different organizations, have one feature in common: Contributions, uploaded by users are associated with the user and the ownership does not change. There is an explicit relationship between the observation (resource) and the user. Expressed as ownership, users can undertake certain actions on their resources.

With the SensorThings API, observations are linked to a data stream that is linked to a sensor which belongs to a thing. The data model does not provide a class that supports explicitly linking a user to an observation. This limits the use of the STA data model and API to a simple use case: One operator can create data objects and all other users have read-only access. This limits the options for applications to interact with the API.

Even though the STA v1.1 data model offers the generic use of `properties`, expressing ownership buried in properties is not wise. Storing personal data in those `properties` makes compliance with the European General Data Protection Regulation (GDPR) difficult. This is because it is unclear if properties store personal data and therefore fall under GDPR. Also, this decreases interoperability tremendously as an application developer would need to know which attribute can be used to express ownership. Lastly, querying observations based on unstructured properties is extremely complex and difficult.

Therefore, the STAplus extension defines the class Party for expressing the association from a `Datastream` or `MultiDatastream` to a user. All uploaded observations, associated to a `Datastream` instance belong to the party associated to the `Datastream`. Parties can also own Things and ObservationGroups, and participate in Campaigns.

6.2. Improving F A I R

In general, there are many aspects when it comes to ensuring the reusability of (existing) data. Licensing is a fundamental aspect, not just in the context of Citizen Science. Very many users contribute to Citizen Science with the motivation of doing meaningful things for the common good. At the same time, users like to be credited when it comes to re-use of their contribution(s). Therefore, most contributions in Citizen Science are freely accessible (open access) and the re-use is “open” but restrictions may apply. In order to get credited, users might associate a license like CC-BY (Creative Commons Attribution License). Even though the data is still freely accessible, there is a condition that must be followed as expressed in the license.

6.3. Licensing

Applying licensing is a difficult task. In particular in a deployment like Citizen Science, where users may decide which license gets applied to their contributions. Too many different and perhaps incompatible licenses exist which may risk a proper re-use. STAplus supports licensing in a flexible way by only demanding that a license has unique URI based identifier (`definition` property). Even though STAplus does not constraint which license types to use, an implementation should constraint the proliferation of licenses via the Business Logic. As illustrated in the OGC Best Practice for using SensorThings API with Citizen Science, an implementation may create all `License` entities and disallow that users create their own licenses. For the Creative Commons licensing framework, this leads to the existence of seven recommended licenses which ensures a proper re-use as license compatibility can easily be computed.

When it comes to attribution licenses, the re-use requires the provision of a citation to name the original author. For the Creative Commons licensing framework, best practices exist that do not need any additional consideration in the STAplus data model. However, a user may want to be able to express an individual attribution to be used with the license. Such information — the actual attribution text — could be stored in the `properties` of the `License` entity. But this would require that the business logic allows creation of `License` entities. To make sure that still only a particular licensing framework can be used, the Business Logic could allow a user to create a `License` entity and store the individual citation into a property like `attributionText` but constrain the `definition` property to those URIs that include the attribution license building block.

6.4. Creating Observation Groups

When contributing to Citizen Science, the relevant information is often a set or bag of individual observations that belong together — i.e., may belong to the same observation event. For example, a camera trap event consists of a picture, a textual observation expanding the likelihood of species prediction and sensor readings for environmental context (temperature, humidity, luminance, air pressure, GPS location). All of these observations potentially created by different sensors could be grouped when created at the same time/location.

Another use case for applying grouping to existing observations is to create a package of observations for the purpose of building the fundamentals for research or to be used in workflows. For researching and later evaluation of a particular phenomenon, the same group of observations could be exchanged via the grouping concept.

Also, over time a user community might link other observations and even provide cross links to other databases. These can be semantically tagged with the `Relation` entity which is described next in this document.

The STAplus extension defines a flexible grouping concept by adding the `ObservationGroup` entity to the STA data model.

6.5. Expressing Relations

Supporting search based on explicit relations is important for Citizen Science. While the STA `ObservedProperty` entity supports links to external resources, the `Observation` entity does not. The STAplus `Relation` entity adds external linking support to `Observation` entities.

The `Relation` entity supports creating generic "– to" relation from an `Observation` entity. The "to" can point to another observation or to any external resource. This allows the generic expression of meaning, leveraging existing semantic concepts that exist elsewhere, for example Dublin or Darwin Core. Adding relations to an `ObservationGroup` entity helps to reason how observations are related even if they were not observed in a same observation event, but instead were linked later to a particular community process (as linking to other databases).

The use of `Relation` entities as defined in STAplus can be applied to already existing SensorThings deployments to semantically enrich the data.

6.6. Data Model Extension

The STAplus data model extension allows expressing the following additional characteristics.

- Party: The `Party` entity supports linking a user to a `Datastream`, `MultiDatastream`, `Campaign` or `ObservationGroup` and `Thing`.
- License: The `License` entity supports expressing reuse conditions by linking a `License` to a `Datastream`, `MultiDatastream`, and / or to a `ObservationGroup`. A `License` on a `Datastream` has the result that all `Observation` entities of that `Datastream` or `MultiDatastream` (as well as entities of the `Thing`, `Sensor`, and `ObservedProperty`) have the associated license. A `License` on an `ObservationGroup` gives the bag or set of

Observations (represented by the ObservationGroup) a license for reuse. Note that the license for each Observation must still be followed.

- **U**nion: The ObservationGroup entity supports packaging individual Observations as a bag or set either as deep copy or via linking
- **S**emantics: The Relation entity supports expressing relationships between Observation entities using the "-to" role type. It is also possible to create relations between Observation entities and external resources using the URI scheme. This allows in particular expressing a relation to any entity of the database (via their external URI). Relation entities can exist on their own or be included into an ObservationGroup to enrich a bag or set of Observation entities.
- Campaign: The Campaign entity is a container of Datastream, MultiDatastream and ObservationGroup entities that supports organizing a campaign or project and to provide metadata as well as legal information such as terms of use and a privacy statement, in case it is relevant.

7

STAPLUS ENTITY TYPE REQUIREMENTS

7 STAPLUS ENTITY TYPE REQUIREMENTS

`Party`, `License`, `ObservationGroup`, `Relation`, and `Campaign` are the STAplus entity types.

In this section, the properties for each STAplus entity type and the direct relation to other entity types are explained.

NOTE: Please be aware that all of the Requirements Classes in this section are part of the Conformance Class Clause 2.2.1.

7.1. Requirements Class Entity Control Information

NOTE: For more information, please see OGC SensorThings API Part 1: Sensing Version 1.1, §8.1.

REQUIREMENTS CLASS 1: ENTITY CONTROL INFORMATION	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
PREREQUISITE	OGC SensorThings API Part 1: Sensing or OGC SensorThings API Part 1: Sensing Version 1.1
NORMATIVE STATEMENT	Requirement 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/common-control-information`

REQUIREMENT 1	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/common-control-information`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
STATEMENT	Each STAplus entity SHALL have the following common control information listed in Table 1.

Table 1 — Common control information

NAME	DEFINITION	DATA TYPE	MULTIPLICITY AND USE
id	`id` is the system-generated identifier of an entity that is unique among the entities of the same entity type in a SensorThings API service instance.	Any	One (mandatory)
selfLink	`selfLink` is the absolute URL of an entity that is unique among all other entities.	URL	One (mandatory)
navigationLink	`navigationLink` is the relative or absolute URL that retrieves content of related entities.	URL	One-to-many (mandatory)

7.2. Requirements Class Party

The `Party` entity can be used to represent acting users and to model ownership.

One example for ownership is that a satellite `Thing` is owned by a space agency. This ownership may entitle the space agency to be the only party that can update the thing's location. Other parties can then mount their sensor on the satellite and provide `Datastream`, or `MultiDatastream` entities to upload observations generated by their sensors. Via the association to the `Datastream` resp. `MultiDatastream` their ownership of the observations is guaranteed. By associating a license to the (multi)datastream, they could also define re-use conditions.

REQUIREMENTS CLASS 2: PARTY	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/party`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
PREREQUISITE	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
NORMATIVE STATEMENTS	Requirement 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/party/properties` Requirement 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/party/relations`

REQUIREMENT 2	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/party/properties`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/party`
STATEMENT	Each `Party` entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 2.

Table 2 — Properties of a `Party` entity

NAME	DEFINITION	DATA TYPE	CARDINALITY AND USE
role	This is the role of the party	CharacterString ['individual', 'institutional']	One (mandatory)
description	This is a short description of the party	CharacterString	Zero or One (optional)
displayName	A property commonly used for saluting the party	CharacterString	Zero or One (optional)
authId	A system wide unique property (e.g., `REMOTE_USER` or `SUB` from authentication) to uniquely identify the party	CharacterString	Zero or One (optional)

REQUIREMENT 3	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/party/relations`
INCLUDED IN	Requirements class 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/party`
STATEMENT	An implementation SHALL allow each `Party` to have direct associations to other entity types listed in Table 3 and Table 4.

Table 3 — Direct relation from a `Party` entity to other entity types

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
Datastreams	Datastream	Zero to Many	A `Party` has zero-to-many `Datastreams`.
MultiDatastreams (optional)	MultiDatastream	Zero to Many	A `Party` has zero-to-many `MultiDatastreams`.

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
Things	Thing	Zero to Many	A Party has zero-to-many Things.
ObservationGroups	ObservationGroup	Zero to Many	A Party has zero-to-many ObservationGroups.
Campaigns	Campaign	Zero to Many	A Party has zero-to-many Campaigns.

Table 4 — Direct relation from a SensorThings entity type to `Party`

RELATION NAME	SOURCE ENTITY TYPE	CARDINALITY	DESCRIPTION
Party	Datastream	Zero or One	A Datastream has zero-to-one Party.
Party (optional)	MultiDatastream	Zero or One	A MultiDatastream has zero-to-one Party.
Party	Thing	Zero or One	A Thing has zero-to-one Party.

7.3. Requirements Class License

The `License` entity can be used to associate a re-use condition to observations via a `Datastream` or `MultiDatastream`. It can also be used to express re-use conditions for an `ObservationGroup` or `Campaign`.

REQUIREMENTS CLASS 3: LICENSE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/license`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
PREREQUISITE	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
NORMATIVE STATEMENTS	Requirement 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/license/properties` Requirement 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/license/relations`

REQUIREMENT 4	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/license/properties`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/license`
STATEMENT	Each `License` entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 5.

Table 5 — Properties of a `License` entity

NAME	DEFINITION	DATA TYPE	CARDINALITY AND USE
name	A property provides a label for `License` entity, commonly a descriptive name.	CharacterString	One (mandatory)
description	This is a short description of the corresponding `License` entity.	CharacterString	Zero or One (optional)
definition	This is a URI referencing the `License` entity.	URI	One (mandatory)
logo	This is the URI of the logo for the `License` entity.	CharacterString	Zero or One (optional)
attributionText	The text to be used as attribution when mandated by the license.	CharacterString	Zero or One (optional)

REQUIREMENT 5	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/license/relations`
INCLUDED IN	Requirements class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/license`
STATEMENT	An implementation SHALL allow each `License` to have direct associations to other entity types listed in Table 6 and Table 7.

Table 6 — Direct relation from a `License` entity to other entity types

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
Datastreams	Datastream	Zero to Many	A `License` has zero-to-many `Datastreams`.
MultiDatastreams (optional)	MultiDatastream	Zero to Many	A `License` has zero-to-many `MultiDatastreams`.

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
Campaigns	Campaign	Zero to Many	A `License` has zero-to-many `Campaigns`.
ObservationGroups	ObservationGroup	Zero to Many	A `License` has zero-to-many `ObservationGroups`.

Table 7 — Direct relation from a SensorThings entity type to `License`

RELATION NAME	SOURCE ENTITY TYPE	CARDINALITY	DESCRIPTION
License	Datastream	Zero or One	A `Datastream` MAY have zero-to-one `License`.
License (optional)	MultiDatastream	Zero or One	A `MultiDatastream` MAY have zero-to-one `License`.

7.4. Requirements Class ObservationGroup

The `ObservationGroup` entity is a bag of observations and/or relations that can be shared and re-used.

REQUIREMENTS CLASS 4: OBSERVATIONGROUP	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/group`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
PREREQUISITE	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
NORMATIVE STATEMENTS	Requirement 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/group/properties` Requirement 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/group/relations`

REQUIREMENT 6	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/group/properties`

REQUIREMENT 6	
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/group`
STATEMENT	Each `ObservationGroup` entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 8.

Table 8 — Properties of a `ObservationGroup` entity

NAME	DEFINITION	DATA TYPE	CARDINALITY AND USE
name	A property provides a label for `ObservationGroup` entity, commonly a descriptive name.	CharacterString	One (mandatory)
description	This is a short description of the corresponding `ObservationGroup` entity.	CharacterString	One (mandatory)
purpose	This is a short description of the purpose for the `ObservationGroup` entity.	CharacterString	Zero or One (optional)
creationTime	This is the time when the `ObservationGroup` was created.	TM Instant	One (mandatory)
endTime	This is the end time of the `ObservationGroup` entity. The exact semantics depend on the business logic.	TM Instant	Zero or One (optional)
termsOfUse	Express the term of use for the `ObservationGroup` entity.	CharacterString	Zero or One (optional)
privacyPolicy	Express the term of use for personal data that are contained in the `ObservationGroup` entity.	CharacterString	Zero or One (optional)
dataQuality	The quality information of the observations in the group	JSON Object	Zero or One (optional)
properties	The SensorThings API definition applies	JSON Object	Zero or One (optional)

REQUIREMENT 7	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/group/relations`
INCLUDED IN	Requirements class 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/group`
STATEMENT	An implementation SHALL allow each `ObservationGroup` to have direct associations to other entity types listed in Table 9 and Table 10.

Table 9 — Direct relation from a `ObservationGroup` entity to other entity types

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
License	License	Zero or One	A `ObservationGroup` has zero-to-one `License`.
Observations	Observation	Zero to Many	A `ObservationGroup` has zero-to-many `Observations`.
Relations	Relation	Zero to Many	A `ObservationGroup` has zero-to-many `Relations`.
Party	Party	Zero or One	A `ObservationGroup` has zero-to-one `Party`.
Campaigns	Campaign	Zero to Many	A `ObservationGroup` has zero-to-many `Campaigns`.

Table 10 — Direct relation from a SensorThings entity type to `ObservationGroup`

RELATION NAME	SOURCE ENTITY TYPE	CARDINALITY	DESCRIPTION
ObservationGroups	Observation	Zero to Many	An `Observation` has zero-to-many `ObservationGroups`.

7.5. Requirements Class Relation

The `Relation` entity can either be used to describe relationships between two observations, or one observation and a resolvable external resource identified by a URI.

REQUIREMENTS CLASS 5: RELATION	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/relation`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
PREREQUISITE	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
NORMATIVE STATEMENTS	Requirement 8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/relation/properties` Requirement 9: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/relation/relations`

REQUIREMENT 8	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/relation/properties`
INCLUDED IN	Requirements class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/relation`
STATEMENT	Each `Relation` entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 11.

Table 11 — Properties of a `Relation` entity

NAME	DEFINITION	DATA TYPE	CARDINALITY AND USE
role	This URI references the definition of the `Relation` entity.	URI	One (mandatory)
description	This is a short description of the corresponding `Relation` entity.	CharacterString	Zero or One (optional)
externalResource	This URI references the external resource for the `Relation` entity.	URI	Zero or One (optional)
properties	The SensorThings API definition applies	JSON Object	Zero or One (optional)

NOTE: The subject of a relation entity is always an observation. For expressing the object of a relation, the `Object` relation XOR `externalResource` property must be used.

REQUIREMENT 9	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/relation/relations`
INCLUDED IN	Requirements class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/relation`
STATEMENT	An implementation SHALL allow each `Relation` to have direct associations to other entity types listed in Table 12.

Table 12 — Direct relation from a `Relation` entity to other entity types

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
Subject	Observation	Zero or One	A `Relation` SHALL have one `Subject`.

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
Object	Observation	Zero or One	A `Relation` has zero-to-one `Object`.
ObservationGroups	ObservationGroup	Zero to Many	A `Relation` has zero-to-many `ObservationGroups`.

NOTE: For expressing the object of a relation, the following constrain applies: `Object` relation XOR `externalResource` property.

Table 13 — Direct relation from a SensorThings entity type to `Relation`

RELATION NAME	SOURCE ENTITY TYPE	CARDINALITY	DESCRIPTION
Objects	Observation	Zero to Many	An `Observation/Subject` has zero-to-many `Relation/Objects`.
Subjects	Observation	Zero to Many	An `Observation/Object` has zero-to-many `Relation/Subjects`.

7.6. Requirements Class Campaign

The `Campaign` entity is a container of `Datastream` or `MultiDatastream` entities. A Campaign can have a particular purpose and a managing party.

REQUIREMENTS CLASS 6: CAMPAIGN	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/campaign`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
PREREQUISITE	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
NORMATIVE STATEMENTS	Requirement 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/campaign/properties` Requirement 11: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/campaign/relations`

REQUIREMENT 10	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/campaign/properties`
INCLUDED IN	Requirements class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/campaign`
STATEMENT	Each `Campaign` entity SHALL have the mandatory properties and MAY have the optional properties listed in Table 14.

Table 14 — Properties of a `Campaign` entity

NAME	DEFINITION	DATA TYPE	CARDINALITY AND USE
name	A property provides a label for `Campaign` entity, commonly a descriptive name.	CharacterString	One (mandatory)
description	This is a short description of the corresponding `Campaign` entity.	CharacterString	One (mandatory)
classification	Determines if the data stream(s), multi data stream (s) or group(s) of the `Campaign` entity contain sensitive information.	ValueCode	Zero or One (optional)
termsOfUse	Express the term of use for the `Campaign` entity.	CharacterString	One (mandatory)
privacyPolicy	Express the terms of use for personal data that are contained in the `Campaign` entity.	CharacterString	Zero or One (optional)
url	This is the URL for the `Campaign` entity that provides additional information that cannot be captured in this entity alone.	URL	Zero or One (optional)
creationTime	This is the time when the `Campaign` entity was created.	TM Instant	One (mandatory)
startTime	This is the starting time of the `Campaign` entity. The exact semantics depend on the business logic.	TM Instant	Zero or One (optional)
endTime	This is the ending time of the `Campaign` entity. The exact semantics depend on the business logic.	TM Instant	Zero or One (optional)

REQUIREMENT 11	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/campaign/relations`
INCLUDED IN	Requirements class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/campaign`

REQUIREMENT 11	
STATEMENT	An implementation SHALL allow each `Campaign` to have direct associations to other entity types listed in Table 15 and Table 16.

Table 15 — Direct relation from a `Campaign` entity to other entity types

RELATION NAME	ENTITY TYPE	CARDINALITY	DESCRIPTION
Datastreams	Datastream	Zero to Many	A `Campaign` has zero-to-many `Datastreams`.
MultiDatastreams (optional)	MultiDatastream	Zero to Many	A `Campaign` has zero-to-many `MultiDatastreams`.
Party	Party	Zero or One	A `Campaign` has zero-to-one `Party`.
ObservationGroups	ObservationGroup	Zero to Many	A `Campaign` has zero-to-many `ObservationGroup`.
License	License	Zero or One	A `Campaign` MAY have zero-to-one `License`.

Table 16 — Direct relation from a SensorThings entity type to `Campaign`

RELATION NAME	SOURCE ENTITY TYPE	CARDINALITY	DESCRIPTION
Campaigns	Datastream	Zero to Many	A `Datastream` has zero-to-many `Campaigns`.
Campaigns (optional)	MultiDatastream	Zero to Many	A `MultiDatastream` has zero-to-many `Campaigns`.

8

STAPLUS READ, CREATE, UPDATE AND DELETE REQUIREMENTS

8 STAPLUS READ, CREATE, UPDATE AND DELETE REQUIREMENTS

8.1. Overview

As many IoT devices are resource-constrained, the SensorThings API adopts the efficient REST web service style. That means the Read, Create, Update, Delete actions can be performed on a STAplus entity type.

8.2. Requirements Class Read

REQUIREMENTS CLASS 7: READ	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/read`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
PREREQUISITE	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
NORMATIVE STATEMENT	Requirement 12: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/read/entity`

REQUIREMENT 12	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/read/entity`
INCLUDED IN	Requirements class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/read`
STATEMENT	To read a STAplus entity or a collection of STAplus entities, the client SHALL send a HTTP GET request to that entity or the collection's URL. If the target URL for the collection is a navigationLink, the entity is automatically linked to the entity or entities represented by the navigationLink. Upon successful completion, the response SHALL contain the representation of the entity or entities.

8.3. Requirements Class Create

REQUIREMENTS CLASS 8: CREATE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/create`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/create`
PREREQUISITE	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information`
NORMATIVE STATEMENTS	Requirement 13: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/entity` Requirement 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/link-to-existing-entities` Requirement 15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/deep-insert` Requirement 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/deep-insert-status-code`

REQUIREMENT 13	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/entity`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/create`
STATEMENT	To create a STAplus entity in a collection, the client SHALL send a HTTP POST request to that collection's URL. The POST body SHALL contain a single valid entity representation. If the target URL for the collection is a navigationLink, the new entity is automatically linked to the entity containing the navigationLink. Upon successful completion, the response SHALL contain a HTTP location header that contains the selfLink of the created entity. Upon successful completion the service SHALL respond with either 201 Created, or 204 No Content. Adapted from OData Version 4.01. Part 1: Protocol, §11.4.2 Create an Entity

REQUIREMENT 14	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/link-to-existing-entities`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/create`
STATEMENT	A STAplus implementation that supports entity creation SHALL support linking new STAplus entities to existing entities upon creation. To create a new entity with links to existing entities in a single request, the client SHALL include the unique identifiers of the related entities associated with the corresponding navigation properties in the request body.

REQUIREMENT 15	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/deep-insert`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/create`
STATEMENT	A request to create a STAplus entity that includes related entities, represented using the appropriate inline representation, is referred to as a “deep insert”. A STAplus implementation that supports entity creation SHALL support deep insert. If the inline representation contains a value for a computed property (*i.e.*, id), the service SHALL ignore that value when creating the related entity. On success, the service SHALL create all entities and relate them. On failure, the service SHALL NOT create any of the entities. Adapted from OData Version 4.01. Part 1: Protocol, §11.4.2.2 Create Related Entities When Creating an Entity

REQUIREMENT 16	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/deep-insert-status-code`
INCLUDED IN	Requirements class 8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/create`
STATEMENT	Upon successfully creating a STAplus entity, the service response SHALL contain a Location header that contains the URL of the created entity. Upon successful completion the service SHALL respond with 201 Created.

8.4. Requirements Class Update

REQUIREMENTS CLASS 9: UPDATE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/update`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update`
PREREQUISITES	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information` https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-update-entity https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-update-entity-put https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-update-entity-jsonpatch
NORMATIVE STATEMENTS	Requirement 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity` Requirement 18: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity-put` Requirement 19: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity-jsonpatch`

REQUIREMENT 17	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity`
INCLUDED IN	Requirements class 9: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/update`
STATEMENT	To update a STAplus entity in a collection a STAplus implementation SHALL follow the requirements as defined in https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-update-entity.

REQUIREMENT 18	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity-put`
INCLUDED IN	Requirements class 9: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/update`
STATEMENT	A STAplus implementation that supports updates with PUT SHALL follow the requirements as defined in https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-update-entity.

REQUIREMENT 19	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity-jsonpatch`
INCLUDED IN	Requirements class 9: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/update`
STATEMENT	A STAplus implementation that supports updates with the JSON PATCH format SHALL follow the requirements as defined in https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-update-entity-jsonpatch

8.5. Requirements Class Delete

REQUIREMENTS CLASS 10: DELETE	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/delete`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/delete`
PREREQUISITES	Requirements class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/entity-control-information` https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-delete-entity
NORMATIVE STATEMENT	Requirement 20: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/delete/entity`

REQUIREMENT 20	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/delete/entity`
INCLUDED IN	Requirements class 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/delete`
STATEMENT	To delete a STAplus entity in a collection a STAplus implementation SHALL follow the requirements as defined in https://docs.ogc.org/is/18-088/18-088.html#req-create-update-delete-delete-entity.

9

STAPLUS MQTT REQUIREMENTS

9 STAPLUS MQTT REQUIREMENTS

The MQTT capability defined in this Standard allows that a client receives changes on STAplus entities via MQTT.

NOTE: The ability for the creation of observations via MQTT is inherited from the SensorThings API Part 1: Sensing and Sensing v1.1 Standard.

9.1. Overview

In the context of MQTT, all STAplus entities as defined in this Standard are equivalent to the STA entities. Therefore, the implementation of MQTT capabilities must be compliant with the SensorThings API Part 1: Sensing and Sensing v1.1 Standard.

9.2. Requirements Class MQTT Subscribe STA v1.0

REQUIREMENTS CLASS 11: MQTT SUBSCRIBE STA V1.0	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/mqtt-subscribe-sta-10`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance test 8-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-10`
PREREQUISITE	http://www.opengis.net/spec/iot_sensing/1.0/req/receive-updates-via-mqtt
NORMATIVE STATEMENT	Requirement 21: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/mqtt-subscribe-sta-10`

REQUIREMENT 21	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/mqtt-subscribe-sta-10`

REQUIREMENT 21	
INCLUDED IN	Requirements class 11: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/mqtt-subscribe-sta-10`
STATEMENT	For the entities defined in STAplus, a STAplus implementation SHALL support the receiving of entity updates with MQTT Subscribe as defined in http://www.opengis.net/spec/iot_sensing/1.0/req/receive-updates-via-mqtt.

9.3. Requirements Class MQTT Subscribe STA v1.1

REQUIREMENTS CLASS 12: MQTT SUBSCRIBE STA V1.1	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/mqtt-subscribe-sta-11`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance test 9-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-11`
PREREQUISITE	http://www.opengis.net/spec/iot_sensing/1.1/req/receive-updates-via-mqtt
NORMATIVE STATEMENT	Requirement 22: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/mqtt-subscribe-sta-11`

REQUIREMENT 22	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/mqtt-subscribe-sta-11`
INCLUDED IN	Requirements class 12: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/mqtt-subscribe-sta-11`
STATEMENT	For the entities defined in STAplus, a STAplus implementation SHALL support the receiving of entity updates with MQTT Subscribe as defined in http://www.opengis.net/spec/iot_sensing/1.1/req/receive-updates-via-mqtt.

10

STAPLUS BUSINESS LOGIC REQUIREMENTS

10 STAPLUS BUSINESS LOGIC REQUIREMENTS

10.1. Overview

The STAplus extension is defined based on requirements from the Citizen Science community. One major requirement from the Citizen Science community is that an implementation of STAplus can be used by many users simultaneously to read, create, update, and delete observations. To ensure integrity of all entities and to prevent inconsistencies in the observations, it is important that an implementation is not only compliant with this Standard but also has functionality that ensures entity consistency.

Even though the business logic can be very complex and most likely depend on many factors, it should be possible providing a hint to developers and guiding end users of the implementation in case a request results in an unexpected response.

As a machine readable and understandable description of the implemented business logic is preferred, the idea of the business logic requirements class is to allow the description of the logic in English text.

10.2. Requirements Class Business Logic

REQUIREMENTS CLASS 13: BUSINESS LOGIC	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/business-logic`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic`
NORMATIVE STATEMENTS	Requirement 23: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/business-logic/definition` Requirement 24: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/business-logic/location`

REQUIREMENT 23

IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/business-logic/definition`
INCLUDED IN	Requirements class 13: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/business-logic`
STATEMENT	The implementation's business logic SHALL be described in English text.

REQUIREMENT 24

IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/business-logic/location`
INCLUDED IN	Requirements class 13: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/business-logic`
STATEMENT	The STAplus implementation SHALL provide a JSON object in the `serverSettings` object on the root URI with the name `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic` that contains a property `href` which value is the URL of the HTML page describing the business logic.

11

STAPLUS AUTHENTICATION REQUIREMENTS

11 STAPLUS AUTHENTICATION REQUIREMENTS

11.1. Overview

To regulate access to entities using CRUD operations, implementations may wish to use information from user / client application authentication to make access control decisions.

In addition, some implementations may wish to link the authentication identifier (e.g., `REMOTE_USER`) with the `Party/authId`. Therefore, the STAplus authentication must provide service-wide unique user identifiers.

11.2. Requirements Class Authentication

REQUIREMENTS CLASS 14: AUTHENTICATION	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/authentication`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`
PREREQUISITE	Requirements class 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/party`
NORMATIVE STATEMENTS	Requirement 25: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id` Requirement 27: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-create` Requirement 26: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-read` Requirement 28: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-update` Requirement 29: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-delete`

REQUIREMENT 25	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id`
INCLUDED IN	Requirements class 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/authentication`
STATEMENT	The STAplus authentication SHALL provide a user identifier permanent for same user but different for each user.

REQUIREMENT 26	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-read`
INCLUDED IN	Requirements class 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/authentication`
STATEMENT	The STAplus implementation SHALL ensure that the `Party/authId` property is exposed in any response and can be used in STA operators such as `$filter` or `$select`.

REQUIREMENT 27	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-create`
INCLUDED IN	Requirements class 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/authentication`
STATEMENT	The STAplus implementation SHALL overwrite any existing value and SHALL set the `Party/authId` property with the user's identification value provided by the authentication (e.g., `REMOTE_USER`).

REQUIREMENT 28	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-update`
INCLUDED IN	Requirements class 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/authentication`
STATEMENT	The STAplus implementation SHALL ensure that the `Party/authId` property cannot be changed (ensure read-only).

REQUIREMENT 29	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-delete`
INCLUDED IN	Requirements class 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/authentication`
STATEMENT	The STAplus implementation SHALL ensure that the `Party/authId` property cannot be deleted.

12

STAPLUS FEATURE AND LOCATION GEOMETRY ENCODING REQUIREMENTS

12 STAPLUS FEATURE AND LOCATION GEOMETRY ENCODING REQUIREMENTS

12.1. Overview

The `Location` and `FeatureOfInterest` entities enable storing data objects with unspecified structure. Both entities potentially store geometry information. The STAplus extension supports multi-user interactions including the creation, updating, reading, and deletion of information. For example in Citizen Science, one service endpoint could accept that users can upload data using different applications. These applications might be developed by different parties and serve different purposes, and thus use different geometry structures and coordinate reference systems (CRS) to encode the coordinates of the geometries. This can result in the situation where the interoperability of uploaded geometry in the `Location` and `FeatureOfInterest` entities might become an issue.

For an implementation of STAplus, this situation becomes complicated when a `$filter` is leveraged that includes spatial conditions such as `ST_Within`. The filter expression does not carry any CRS information. Therefore, how does the implementation “know” how to apply a spatial filter to geometries stored in `feature` and `location` properties uploaded by different applications? Because the SensorThings data model does not define how to store CRS information inside the `feature` and `location`, it only recommends using GeoJSON. So, applying the `$filter` with spatial operators introduces two problems. First, the CRS data would need to be stored in a standardized location inside `feature` and `location` properties. Second, the implementation would need to apply coordinate transformation ‘on the fly’ when processing a spatial filter condition and stored geometries are encoded in different CRS. The first problem causes interoperability issues and the later inevitably causes performance issues.

To get out of this situation, the STAplus Standard defines a default-CRS based on WGS84 with axis-order longitude/latitude as defined in GeoJSON. Any uploaded geometry data encoded in the default-CRS will be stored as-is. Any uploaded geometry data encoded differently will be rejected.

To indicate that the default-CRS and GeoJSON geometry encoding is used, the `encodingType` property of the `FeatureOfInterest` and `Location` entity is to be used with the value `application/geo+json`.

In addition, the STAplus extension supports other (commonly used) encodings via the `Geometry Encoding` Requirements Class.

12.2. Default-CRS Requirements Class

This Requirements Class defines the default-CRS with axis-order and its media-type.

REQUIREMENTS CLASS 15: DEFAUL-CRS	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/default-crs`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
PREREQUISITE	GeoJSON
NORMATIVE STATEMENTS	Requirement 30: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/crs-definition` Requirement 31: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/axis-order` Requirement 32: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/media-type` Requirement 33: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/processing`

REQUIREMENT 30	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/crs-definition`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/default-crs`
STATEMENT	The STAplus implementation SHALL use the GeoJSON CRS `urn:ogc:def:crs:OGC::CRS84` as the default-CRS.

REQUIREMENT 31	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/axis-order`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/default-crs`
STATEMENT	The STAplus implementation SHALL use the GeoJSON axis-order with the default-CRS.

REQUIREMENT 32	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/media-type`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/default-crs`
STATEMENT	The STAplus implementation SHALL accept the media-type `application/geo+json` value for the `encodingType` property to indicate that the structuring of the `FeatureOfInterest` and `Location` entities geometry encoding is compliant with the RFC GeoJSON.

REQUIREMENT 33	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/processing`
INCLUDED IN	Requirements class 15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/default-crs`
STATEMENT	The STAplus implementation SHALL enforce the `default-CRS` to any geometry data contained in the `FeatureOfInterest` and `Location` entity. The STAplus implementation SHALL reject any geometry **not** encoded in the `default-CRS` without further processing.

12.3. Geometry-FG Requirements Class

NOTE: This Requirements Class remains informative until OGC Features and Geometries JSON - Part 1: Core is an adopted OGC Standard.

This Requirements Class defines the use of geometry encoding compliant with the OGC Draft Standard OGC Features and Geometries JSON - Part 1: Core

REQUIREMENTS CLASS 16: GEOMETRY-FG	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`

REQUIREMENTS CLASS 16: GEOMETRY-FG	
PREREQUISITE	OGC Features and Geometries JSON - Part 1: Core
NORMATIVE STATEMENTS	Requirement 34: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/media-type` Requirement 35: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/default-crs` Requirement 36: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/supported-crs` Requirement 37: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/crs-error` Requirement 38: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/processing` Requirement 39: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/out`

REQUIREMENT 34	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/media-type`
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
STATEMENT	The STAplus implementation SHALL accept the media-type `application/vnd.ogc.fg+json` as value to the `encodingType` property of the `FeatureOfInterest` and `Location` entities to indicate that the structuring of the geometry is be compliant with OGC Features and Geometries JSON - Part 1: Core

REQUIREMENT 35	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/default-crs`
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
STATEMENT	The STAplus implementation SHALL advertise the default-CRS on the conformance page.

REQUIREMENT 36	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/supported-crs`

REQUIREMENT 36	
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
STATEMENT	The STAplus implementation SHALL advertise the list of the supported CRS on the conformance page.

REQUIREMENT 37	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/crs-error`
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
STATEMENT	The STAplus implementation SHALL return an error if the geometry data inside `feature` or `location` properties is encoded in an unsupported CRS.

REQUIREMENT 38	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/processing`
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
STATEMENT	If necessary, the implementation SHALL apply a CRS transformation to the `default-CRS` if necessary before further processing or storing the geometry data.

REQUIREMENT 39	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/out`
INCLUDED IN	Requirements class 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-fg`
STATEMENT	The STAplus implementation SHALL use the default-CRS to encode the `feature` and `location` geometries in a response.

12.4. Geometry WKT Requirements Class

This Requirements Class defines the use of geometry encoding compliant with Well Known Text (WKT).

REQUIREMENTS CLASS 17: GEOMETRY WKT	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
OBLIGATION	requirement
TARGET TYPE	Target Type: Implementation
CONFORMANCE CLASS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
PREREQUISITE	Geographic information - Simple features access - Part 1: Common architecture
NORMATIVE STATEMENTS	Requirement 40: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/media-type` Requirement 17-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/crs-defintion` Requirement 42: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/default-crs` Requirement 43: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/supported-crs` Requirement 44: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/crs-error` Requirement 45: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/value` Requirement 46: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/processing` Requirement 47: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/out`

REQUIREMENT 40	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/media-type`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
STATEMENT	The STAplus implementation SHALL accept the media-type `application/wkt` as value for the `encodingType` property of the `FeatureOfInterest` and `Location` entities. **NOTE:** Media type `application/wkt` is currently pending approval by IANA.

REQUIREMENT 41	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/crs-definition`
STATEMENT	If a non-default-CRS is used then either the CRS identifier SHALL be put into a property `crs`, or the CRS identifier (number) SHALL be put into a property `srid` of the `properties` property of the `FeatureOfInterest` or `Location` entity.

REQUIREMENT 42	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/default-crs`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
STATEMENT	The STAplus implementation SHALL provide a JSON object in the `serverSettings` object on the root URI with the name `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` that contains a property `default-crs` whose value represents the default-CRS identifier.

REQUIREMENT 43	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/supported-crs`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
STATEMENT	The STAplus implementation SHALL provide a JSON object in the `serverSettings` object on the root URI with the name `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` that contains a property `supported-crs` of type Array which values represent the supported CRS identifiers.

REQUIREMENT 44	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/crs-error`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
STATEMENT	The STAplus implementation SHALL return an error if the geometry data inside the `FeatureOfInterest` or `Location` is encoded in an unsupported CRS.

REQUIREMENT 45	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/value`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
STATEMENT	The WKT encoded geometry SHALL be the value of the `feature` or `location` property (the type Any is a String).

REQUIREMENT 46	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/processing`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
STATEMENT	The STAplus implementation SHALL apply CRS transformation to the `default-CRS` if necessary before further processing or storing the geometry data.

REQUIREMENT 47	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/out`
INCLUDED IN	Requirements class 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req-class/geometry-wkt`
STATEMENT	The STAplus implementation SHALL use the default-CRS to encode the `feature` and `location` geometries in a response.

13

MEDIA TYPES FOR FEATUREOFINTEREST AND LOCATION ENCODING

13 MEDIA TYPES FOR FEATUREOFINTEREST AND LOCATION ENCODING

This Standard offers support for different structuring of the `FeatureOfInterest` and `Location` entities. To indicate the support for a particular structure, an implementation can use one of the following media-types:

- `application/geo+json`
- `application/vnd.ogc.fg+json`[1]
- `application/wkt`[2]

[1]Informative until OGC Features and Geometries JSON - Part 1: Core is an adopted OGC Standard.

[2]Pending approval by IANA.

A

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE

A ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE

This normative section defines the STAplus 1.0 Conformance Class tests.

NOTE: A STAplus compliant implementation must also be compliant with the SensorThings API conformance as defined in OGC SensorThings API Part 1: Sensing, respectively OGC SensorThings API Part 1: Sensing Version 1.1.

A.1. STAplus Core Conformance Class Tests

CONFORMANCE TEST A.1	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/common-control-information`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/common-control-information`
INCLUDED IN	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TEST PURPOSE	To verify that the common control information is available as defined in the requirement.
TEST-METHOD-TYPE	Manually Inspect
TEST METHOD	Inspect the full JSON object of the entity sets (*i.e.*, without $select) to identify, if each entity has the common control information defined in Requirement Class Clause 7.1 and the service sends appropriate responses as defined in this Standard.

CONFORMANCE TEST A.2	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/entities`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/party/properties` Requirement 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/party/relations` Requirement 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/license/properties` Requirement 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/license/relations` Requirement 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/group/properties` Requirement 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/group/relations` Requirement 8: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/relation/properties` Requirement 9: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/relation/relations` Requirement 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/campaign/properties` Requirement 11: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/campaign/relations`
INCLUDED IN	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
PREREQUISITE	Conformance test A.1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/common-control-information`
TEST PURPOSE	Verify that each STAplus entity has the mandatory properties and mandatory relations as defined in this Standard.
TEST-METHOD-TYPE	Manually Inspect
TEST METHOD	Evaluate for each STAplus entity:
A	Inspect the full JSON object of the entity sets (*i.e.*, without $select) to identify, if each entity has the mandatory properties defined in the corresponding requirement.
B	Inspect the full JSON object of each entity set (*i.e.*, without using the $select query option) to identify, if each entity has the mandatory relations (*i.e.*, @iot.navigationLink) defined in the corresponding requirement.

CONFORMANCE TEST A.3	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/read`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 12: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/read/entity`
INCLUDED IN	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
PREREQUISITE	Conformance test A.2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/entities`
TEST PURPOSE	Verify that the implementation supports reading STAplus entities via HTTP GET.
TEST-METHOD-TYPE	Manually Inspect
TEST METHOD	Evaluate that the implementation accepts a Sensor Things API compliant HTTP Get request to the STAplus entities:
A	Construct a URL to the `Party` entity and verify the response.
B	Construct a URL to the `License` entity and verify the response.
C	Construct a URL to the `ObservationGroup` entity and verify the response.
D	Construct a URL to the `Relation` entity and verify the response.
E	Construct a URL to the `Campaign` entity and verify the response.

CONFORMANCE TEST A.4	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/crs-definition`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 30: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/crs-definition`
INCLUDED IN	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TEST PURPOSE	Verify that the implementation supports and uses the default-CRS.
TEST METHOD	Evaluate that the implementation uses the default-CRS.

CONFORMANCE TEST A.4	
A	Construct a `Location` entity that contains a `Location` property whose geometry is encoded using the default-CRS and check that the implementation is processing the geometry accordingly and that the geometry data is stored using the default-CRS.
B	Construct a `FeatureOfInterest` entity that contains a `Feature` property whose geometry is encoded using the default-CRS and check that the implementation is processing the geometry accordingly and that the geometry data is stored using the default-CRS.

CONFORMANCE TEST A.5	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/axis-order`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 31: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/axis-order`
INCLUDED IN	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TEST PURPOSE	Verify that the implementation supports and uses the default axis-order.
TEST METHOD	Evaluate that the implementation uses the default axis-order.
A	Construct a `Location` entity that contains a `location` property whose geometry is encoded using the default axis-order and check that the implementation is processing the geometry accordingly and that the geometry data is stored using the default-CRS.
B	Construct a `FeatureOfInterest` entity that contains a `feature` property whose geometry is encoded using the default axis-order and check that the implementation is processing the geometry accordingly and that the geometry data is stored using the default-CRS.

CONFORMANCE TEST A.6	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/media-type`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 32: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/media-type`
INCLUDED IN	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TEST PURPOSE	Verify that the implementation supports and uses the default media-type.

CONFORMANCE TEST A.6	
TEST METHOD	Evaluate that the implementation uses the default media-type.
A	Construct a `Location` entity that contains a `location` property whose geometry is encoded using the default-CRS and axis-order where the `encodingType` property's value is `application/geo+json` and check that the implementation is processing the geometry accordingly and that the geometry data is stored using the default-CRS and axis-order.
B	Construct a `FeatureOfInterest` entity that contains a `feature` property whose geometry is encoded using the default-CRS and axis-order where the `encodingType` property's value is `application/geo+json` and check that the implementation is processing the geometry accordingly and that the geometry data is stored using the default-CRS and axis-order.

CONFORMANCE TEST A.7	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core/default-crs/processing`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 33: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/default-crs/processing`
INCLUDED IN	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core`
TEST PURPOSE	Verify that the implementation stores geometry that is encoded in the default-CRS and axis-order without processing.
TEST METHOD	Evaluate that the implementation stores geometry that is encoded in the default-CRS and axis-order without processing.
A	Construct a `Location` entity that contains a `location` property whose geometry is encoded using the default-CRS and axis-order where the `encodingType` property's value is `application/geo+json` and check that the implementation is stores the geometry data without processing.
B	Construct a `FeatureOfInterest` entity that contains a `feature` property whose geometry is encoded using the default-CRS and axis-order where the `encodingType` property's value is `application/geo+json` and check that the implementation is storing the geometry data without a CRS transformation.

A.2. STAplus Create Conformance Class Tests

CONFORMANCE TEST A.8	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/create/http`
REQUIREMENTS	Conformance class 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/create` Requirement 13: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/entity` Requirement 14: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/link-to-existing-entities` Requirement 15: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/deep-insert` Requirement 16: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/create/deep-insert-status-code`
INCLUDED IN	Conformance class 2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/create`
TEST PURPOSE	To verify that the service implementation supports the creation of entities as defined in this Standard.
TEST METHOD	For each STAplus entity:
A	Create an entity instance by following the integrity constraints and creating the related entities with a single request (*i.e.*, deep insert), check if the entity instance is successfully created and the implementation responds as defined in this Standard.
B	Create an entity instance and its related entities with a deep insert request that does not conform to the Standard (e.g., missing a mandatory property), check if the service fails the request without creating any entity within the deep insert request and responds the appropriate HTTP status code.
C	Issue an entity creation request that does not follow the integrity constraints with deep insert, check if the service fails the request without creating any entity within the deep insert request and responds the appropriate HTTP status code.
D	Create an entity instance by linking to existing entities with a single request, check if the server responds as defined in this Standard.
E	Create an entity instance that does not follow the integrity constraints by linking to existing entities with a single request, check if the server responds as defined in this Standard.

A.3. STAplus Update Conformance Class Tests

CONFORMANCE TEST A.9	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update/put`

CONFORMANCE TEST A.9	
REQUIREMENTS	Conformance class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update` Requirement 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity` Requirement 18: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity-put`
INCLUDED IN	Conformance class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update`
TEST PURPOSE	To verify that the service implementation supports the update of entities as defined in this Standard.
TEST METHOD	For each STAplus entity:
A	Send an update request with HTTP PATCH and check if the service responds as defined.

CONFORMANCE TEST A.10	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update/patch`
REQUIREMENTS	Conformance class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update` Requirement 17: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity` Requirement 19: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/update/entity-jsonpatch`
INCLUDED IN	Conformance class 3: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/update`
TEST PURPOSE	To verify that the service implementation supports the update of entities as defined in this Standard.
TEST METHOD	For each STAplus entity:
A	Send an update request with PATCH, check (1) if the properties provided in the payload corresponding to updatable properties replace the value of the corresponding property in the entity and (2) if the missing properties of the containing entity or complex property are not directly altered.
B	Send an update request with PATCH that contains related entities as inline content, check if the service fails the request and returns appropriate HTTP status code.
C	Send an update request with PATCH that contains binding information for navigation properties, check if the service updates the navigationLink accordingly.

A.4. STAplus Delete Conformance Class Tests

CONFORMANCE TEST A.11	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/delete/entity`
REQUIREMENTS	Conformance class 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/delete` Requirement 20: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/delete/entity`
INCLUDED IN	Conformance class 4: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/delete`
TEST PURPOSE	To verify that the service implementation supports the deletion of entities as defined.
TEST METHOD	For each STAplus entity:
A	Delete an entity instance, and check if the service responds as defined.

A.5. STAplus Authentication Conformance Class Tests

CONFORMANCE TEST A.12	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id`
REQUIREMENTS	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication` Requirement 25: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id`
INCLUDED IN	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`
TEST PURPOSE	To verify that the user's identifier is permanent and unique.
TEST METHOD	Verify the following:
A	Compare the user identifier after repeated login of the same user and verify that the identifier is identical.

CONFORMANCE TEST A.12	
B	Compare the user identifier for different users and verify that the identifiers are different.

CONFORMANCE TEST A.13	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-create`
REQUIREMENTS	Conformance class 1: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/core` Requirement 27: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-create`
INCLUDED IN	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`
TEST PURPOSE	To verify that the identifier value stored in `Party/authId` is the value set by the authentication.
TEST METHOD	For a non-existing `Party` entity:
A	Submit a HTTP POST request to create a `Party` entity where the `authId` is not set.
B	Observe the response to find the location URI for the created `Party` entity.
C	Issue a HTTP GET request to the location URI and verify that the `Party/authId` is set with the unique identifier that represents the user.
D	Submit a HTTP POST request to create a `Party` entity where the `authId` is set with some value.
E	Observe the response status code and verify it is 400.

CONFORMANCE TEST A.14	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-read`
REQUIREMENTS	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication` Requirement 26: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-read`
INCLUDED IN	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`
TEST PURPOSE	To verify that the `Party/authId` is exposed in a response.

CONFORMANCE TEST A.14	
TEST METHOD	Verify that response from any `Party` entity includes the `authId`:
A	Construct a HTTP GET request to some `Party` entity and verify that the `authId` is contained in the response.

CONFORMANCE TEST A.15	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-update`
REQUIREMENTS	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication` Requirement 28: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-update`
INCLUDED IN	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`
TEST PURPOSE	To verify that a user can **not** update the identifier stored in `Party/authId`.
TEST METHOD	Verify that changing the `authId` is not possible for the user itself:
A	Have the user authenticate and identify the corresponding `Party`.
B	Construct a HTTP PATCH request to that **same** `Party` entity where the `authId` is set with some value.
C	Issue a HTTP GET request to that `Party` and verify that the value of the `authId` has not changed.
TEST METHOD	Verify that changing the `authId` is not possible for any other user:
D	Have the user authenticate and identify the corresponding `Party`.
E	Construct a HTTP PATCH request to **another** existing `Party` entity where the `authId` is set with some value.
F	Issue a HTTP GET request to that other `Party` and verify that the value of the `authId` has not changed.

CONFORMANCE TEST A.16	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication/id-delete`
REQUIREMENTS	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`

CONFORMANCE TEST A.16	
	Requirement 29: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/authentication/id-delete`
INCLUDED IN	Conformance class 5: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/authentication`
TEST PURPOSE	To verify that a user can **not** delete the identifier stored in `Party/authId`.
TEST METHOD	Verify that deleting the `authId` is not possible for the user itself:
A	Have the user authenticate and identify the corresponding `Party`.
B	Construct a HTTP DELETE request to that `Party` entity.
C	Issue a HTTP GET request to that `Party` and verify that the entity still exists.
TEST METHOD	Verify that deleting the `authId` is not possible for any other user:
D	Have the user authenticate and identify the corresponding `Party`.
E	Construct a HTTP DELETE request to **another** existing `Party`.
F	Issue a HTTP GET request to that other `Party` and verify that the entity still exists.

A.6. STAplus Business Logic Conformance Class Tests

CONFORMANCE TEST A.17	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic/definition`
REQUIREMENTS	Conformance class 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic` Requirement 23: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/business-logic/definition`
INCLUDED IN	Conformance class 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic`
TEST PURPOSE	To verify that the description of the business logic is human readable and in English.
TEST METHOD	Verify that the HTML page for the business logic is in English language.

CONFORMANCE TEST A.18	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic/location`
REQUIREMENTS	Conformance class 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic` Requirement 24: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/business-logic/location`
INCLUDED IN	Conformance class 10: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic`
TEST PURPOSE	To verify that the business logic is available from the provided URL.
TEST METHOD	On the service root URI, find the JSON object with name `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/business-logic` and follow the link provided in the `href` property. Verify that the loaded HTML page contains the description of the business logic.

A.7. STAplus Geometry FG Conformance Class Tests

NOTE: These Conformance Class Tests remain informative until OGC Features and Geometries JSON - Part 1: Core is an adopted OGC Standard.

CONFORMANCE TEST A.19	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg//media-type`
REQUIREMENTS	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg` Requirement 34: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/media-type`
INCLUDED IN	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`
TEST PURPOSE	To verify that the implementation accepts media-type for Geometry-FG.
TEST METHOD	Verify that the implementation supports the use of the media-type for Geometry-FG.

CONFORMANCE TEST A.20	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/default-crs`
REQUIREMENTS	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg` Requirement 35: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/default-crs`
INCLUDED IN	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`
TEST PURPOSE	To verify that the default-CRS is used for processing geometry data from `Feature` and `Location`.
TEST METHOD	Verify that the implementation applies the default-CRS advertised in the conformance page to the geometry data from `Feature` and `Location`.

CONFORMANCE TEST A.21	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/supported-crs`
REQUIREMENTS	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg` Requirement 36: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/supported-crs`
INCLUDED IN	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`
TEST PURPOSE	To verify that all CRS, advertised as supported in the conformance page are accepted.
TEST METHOD	Verify that the implementation accepts geometry encodings for `Feature` and `Location`. For each supported CRS:
A	Construct a geometry and create a `Location` and `FeatureOfInterest` entity. Verify that the geometry data is accepted by the implementation.

CONFORMANCE TEST A.22	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/crs-error`
REQUIREMENTS	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg` Requirement 37: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/crs-error`

CONFORMANCE TEST A.22	
INCLUDED IN	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`
TEST PURPOSE	To verify that no additional CRS, as advertised in the conformance page, are accepted.
TEST METHOD	Verify that the implementation **does not** accept geometry encodings for `Feature` and `Location` that are not listed as supported. For a CRS **not** listed as supported:
A	Construct a geometry and create a `Location` and `FeatureOfInterest` entity. Verify that the geometry data is **rejected** by the implementation.

CONFORMANCE TEST A.23	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/processing`
REQUIREMENTS	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg` Requirement 38: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/processing`
INCLUDED IN	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`
TEST PURPOSE	To verify that a geometry not encoded in the default-CRS is transformed before storage.
TEST METHOD	Verify that the implementation accepts geometry encodings for `Feature` and `Location` that use a supported CRS:
A	Construct a geometry and create a `Location` and `FeatureOfInterest` entity. Verify that the geometry data is accepted and transformed to the default-CRS before processed and stored by the implementation.

CONFORMANCE TEST A.24	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg/out`
REQUIREMENTS	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg` Requirement 39: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-fg/out`
INCLUDED IN	Conformance class 6: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-fg`
TEST PURPOSE	To verify that a geometry included in a response is encoded in the default-CRS.

CONFORMANCE TEST A.24	
TEST METHOD	Verify that the geometry data for a `Feature` and `Location` is using default-CRS, independent from the geometry CRS used with the creation or updating of the entity.

A.8. STAplus Geometry WKT Conformance Class Tests

CONFORMANCE TEST A.25	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/media-type`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 40: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/media-type`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that the implementation accepts media-type for WKT.
TEST METHOD	Verify that the implementation supports the use of the media-type for WKT.

CONFORMANCE TEST A.26	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/crs-definition`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 41: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/crs-definition`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that the implementation accepts CRS definition provided in the associated property.
TEST METHOD	Verify that the implementation supports the use of the CRS property.
A	Construct a WKT geometry in a CRS different from the `default-crs`.
B	Set the `crs` property to the CRS identifier.

CONFORMANCE TEST A.26	
C	Verify that the implementation processes the geometry honoring the CRS identified by the `crs` value.
D	Set the `srid` property to the CRS identifier number.
E	Verify that the implementation processes the geometry honoring the CRS identified by the `srid` value.

CONFORMANCE TEST A.27	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/default-crs`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 42: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/default-crs`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that the default-CRS is used for processing geometry data from `feature` and `location` property.
TEST METHOD	Verify that the implementation defines and applies the default-CRS to the geometry data from `feature` and `location` property.
A	Find the JSON object in the `serverSettings` object on the root URI with the name `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` and check the value of the property `default-crs`.
B	Verify that the `default-crs` is applied to a WKT geometry if no `crs` or `srid` property is used.

CONFORMANCE TEST A.28	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/supported-crs`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 43: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/supported-crs`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that all supported CRS are accepted.

CONFORMANCE TEST A.28	
TEST METHOD	Verify that the implementation accepts geometry encodings for `feature` and `location` properties. For each supported CRS:
A	Execute test http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/crs-definition and verify that the implementation processes the geometry correctly.

CONFORMANCE TEST A.29	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/crs-error`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 44: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/crs-error`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that only supported CRSs are accepted.
TEST METHOD	Verify that the implementation **does not** accept geometry encodings for `feature` and `location` properties that are not listed as supported. For a CRS **not** listed as supported:
A	Execute test http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/crs-definition and verify that the geometry data is **rejected** by the implementation.

CONFORMANCE TEST A.30	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/value`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 45: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/value`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that the geometry value, compliant to WKT is accepted as value for the `feature` and `location` property.
TEST METHOD	Verify that the implementation accepts WKT geometry values for `feature` and `location` properties.

CONFORMANCE TEST A.31	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/processing`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 46: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/processing`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that a geometry not encoded in the default-CRS is transformed before storage.
TEST METHOD	Verify that the implementation accepts geometry encodings for `Feature` and `Location` that use a supported CRS:
A	Construct a geometry and create a `Location` and `FeatureOfInterest` entity. Verify that the geometry data in the `location` and `feature` properties is accepted and transformed to the default-CRS before processed and stored by the implementation.

CONFORMANCE TEST A.32	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt/out`
REQUIREMENTS	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt` Requirement 47: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/geometry-wkt/out`
INCLUDED IN	Conformance class 7: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/geometry-wkt`
TEST PURPOSE	To verify that a geometry included in a response is encoded in the default-CRS.
TEST METHOD	Verify that the geometry data for a `feature` and `location` properties is using default-CRS, independent from the geometry CRS used with the creation or updating of the entity.

A.9. STAplus MQTT Subscribe STA v1.0 Conformance Class Tests

CONFORMANCE TEST A.33	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-10/definition`
REQUIREMENTS	Conformance test 8-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-10` Requirement 21: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/mqtt-subscribe-sta-10`
INCLUDED IN	Conformance test 8-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-10`
TEST PURPOSE	To verify that a client can receive notifications for the updates of a STAplus entity set or an individual entity with MQTT.
TEST METHOD	For each STAplus entity:
A	Subscribe to an entity set with MQTT Subscribe. Then create a new entity of the subscribed entity set. Check if a complete JSON representation of the newly created entity through MQTT is received.
B	Subscribe to an entity set with MQTT Subscribe. Then update an existing entity of the subscribed entity set. Check if a complete JSON representation of the updated entity through MQTT is received.
C	Subscribe to an entity's property with MQTT Subscribe. Then update the property with PATCH. Check if the JSON object of the updated property is received.
DESCRIPTION	Subscribe to multiple properties of an entity set with MQTT Subscribe. Then create a new entity of the entity set. Check if a JSON object of the subscribed properties is received. part: Subscribe to multiple properties of an entity set with MQTT Subscribe. Then update an existing entity of the entity set with PATCH. Check if a JSON object of the subscribed properties is received.

A.10. STAplus MQTT Subscribe STA v1.1 Conformance Class Tests

CONFORMANCE TEST A.34	
IDENTIFIER	`http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-11/definition`
REQUIREMENTS	Conformance test 9-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-11`

CONFORMANCE TEST A.34	
	Requirement 22: `http://www.opengis.net/spec/sensorthings-staplus/1.0/req/mqtt-subscribe-sta-11`
INCLUDED IN	Conformance test 9-2: `http://www.opengis.net/spec/sensorthings-staplus/1.0/conf/mqtt-subscribe-sta-11`
TEST PURPOSE	To verify that a client can receive notifications for the updates of a STAplus entity set or an individual entity with MQTT.
TEST METHOD	For each STAplus entity:
A	Subscribe to an entity set with MQTT Subscribe. Then create a new entity of the subscribed entity set. Check if a complete JSON representation of the newly created entity through MQTT is received.
B	Subscribe to an entity set with MQTT Subscribe. Then update an existing entity of the subscribed entity set. Check if a complete JSON representation of the updated entity through MQTT is received.
C	Subscribe to an entity's property with MQTT Subscribe. Then update the property with PATCH. Check if the JSON object of the updated property is received.
DESCRIPTION	Subscribe to multiple properties of an entity set with MQTT Subscribe. Then create a new entity of the entity set. Check if a JSON object of the subscribed properties is received. part: Subscribe to multiple properties of an entity set with MQTT Subscribe. Then update an existing entity of the entity set with PATCH. Check if a JSON object of the subscribed properties is received.

B

ANNEX B (INFORMATIVE) STAPLUS DATA MODEL (INFORMATIVE)

B

ANNEX B (INFORMATIVE) STAPLUS DATA MODEL (INFORMATIVE)

This appendix contains the Data Model UML diagrams in large for improved readability.

B.1. STAplus Data Model connected to Datastream

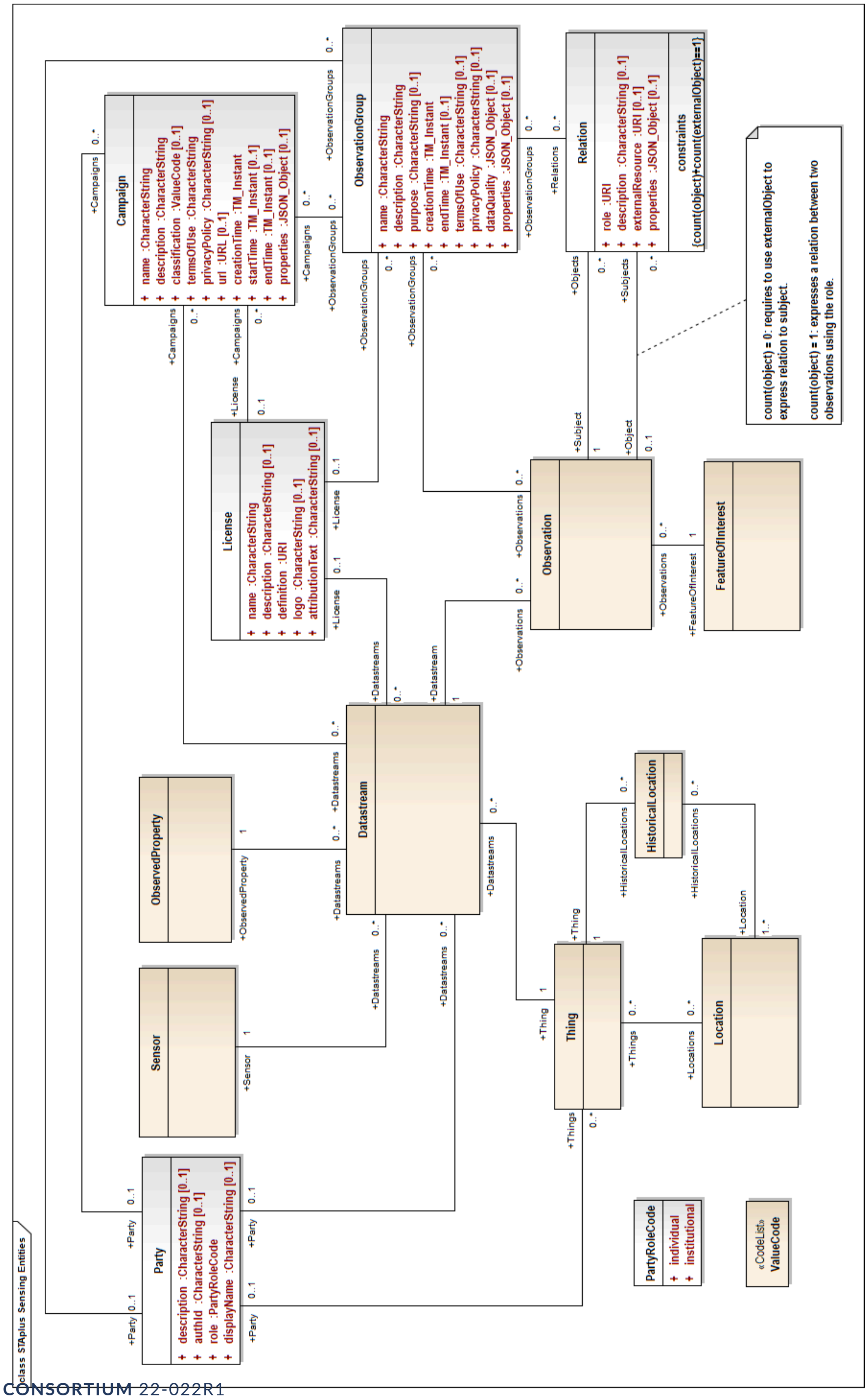
class STAplus Sensing Entities
Campaign
+ name :CharacterString
+ description :CharacterString
+ classification :ValueCode [0..1]
+ termsOfUse :CharacterString
+ privacyPolicy :CharacterString [0..1]
+ url :URL [0..1]
+ creationTime :TM_Instant
+ startTime :TM_Instant [0..1]
+ endTime :TM_Instant [0..1]
+ properties :JSON_Object [0..1]
ObservationGroup
+ name :CharacterString
+ description :CharacterString
+ purpose :CharacterString [0..1]
+ creationTime :TM_Instant
+ endTime :TM_Instant [0..1]
+ termsOfUse :CharacterString [0..1]
+ privacyPolicy :CharacterString [0..1]
+ dataQuality :JSON_Object [0..1]
+ properties :JSON_Object [0..1]
Relation
+ role :URI
+ description :CharacterString [0..1]
+ externalResource :URI [0..1]
+ properties :JSON_Object [0..1]
constraints
{count(object)+count(externalObject)==1}
count(object) = 0: requires to use externalObject to express relation to subject.
count(object) = 1: expresses a relation between two observations using the role.
License
+ name :CharacterString
+ description :CharacterString [0..1]
+ definition :URI
+ logo :CharacterString [0..1]
+ attributionText :CharacterString [0..1]
Observation
FeatureOfInterest
Datastream
ObservedProperty
Sensor
HistoricalLocation
Thing
Location
Party
+ description :CharacterString [0..1]
+ authId :CharacterString [0..1]
+ role :PartyRoleCode
+ displayName :CharacterString [0..1]
PartyRoleCode
+ individual
+ institutional
«CodeList»
ValueCode
+Campaigns 0..*
+ObservationGroups 0..*
+Relations 0..*
+Objects 0..*
+Subjects 0..*
+Subject 1
+Object 0..1
+License 0..1
+Datastreams 0..*
+Datastream 1
+Observations 0..*
+FeatureOfInterest 1
+ObservedProperty 1
+Sensor 1
+Thing 1
+Things 0..*
+Locations 0..*
+Location 1..*
+HistoricalLocations 0..*
+Party 0..1

Figure B.1

B.2. STAplus Data Model connected to MultiDatastream

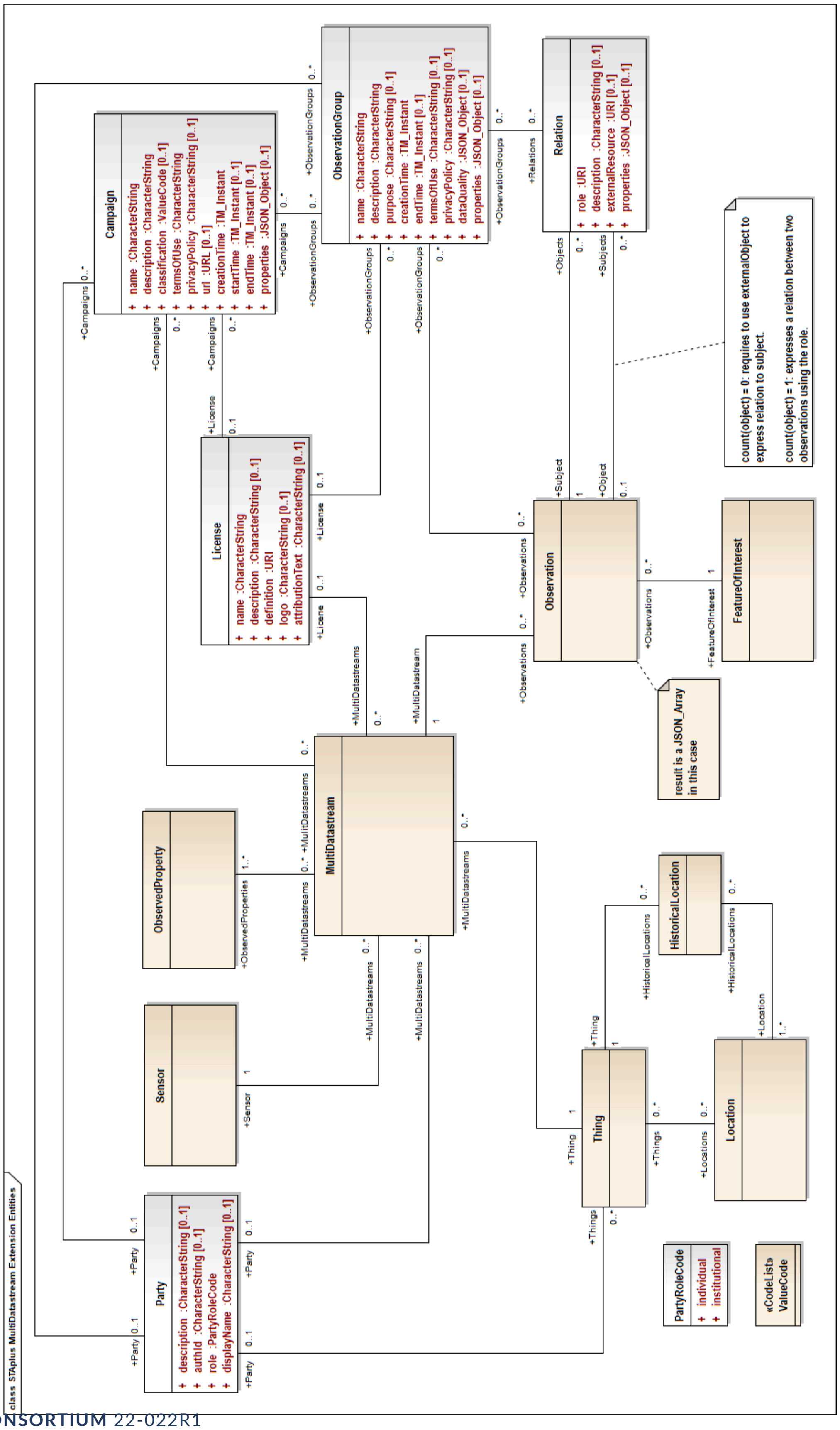
class STAplus MultiDatastream Extension Entities
Campaign
+ name :CharacterString
+ description :CharacterString
+ classification :ValueCode [0..1]
+ termsOfUse :CharacterString
+ privacyPolicy :CharacterString [0..1]
+ url :URL [0..1]
+ creationTime :TM_Instant
+ startTime :TM_Instant [0..1]
+ endTime :TM_Instant [0..1]
+ properties :JSON_Object [0..1]
ObservationGroup
+ name :CharacterString
+ description :CharacterString
+ purpose :CharacterString [0..1]
+ creationTime :TM_Instant
+ endTime :TM_Instant [0..1]
+ termsOfUse :CharacterString [0..1]
+ privacyPolicy :CharacterString [0..1]
+ dataQuality :JSON_Object [0..1]
+ properties :JSON_Object [0..1]
Relation
+ role :URI
+ description :CharacterString [0..1]
+ externalResource :URI [0..1]
+ properties :JSON_Object [0..1]
License
+ name :CharacterString
+ description :CharacterString [0..1]
+ definition :URI
+ logo :CharacterString [0..1]
+ attributionText :CharacterString [0..1]
count(object) = 0: requires to use externalObject to express relation to subject.
count(object) = 1: expresses a relation between two observations using the role.
Observation
FeatureOfInterest
result is a JSON_Array in this case
MultiDatastream
ObservedProperty
Sensor
HistoricalLocation
Thing
Location
Party
+ description :CharacterString [0..1]
+ authId :CharacterString [0..1]
+ role :PartyRoleCode
+ displayName :CharacterString [0..1]
PartyRoleCode
+ individual
+ institutional
«CodeList»
ValueCode
+Campaigns
+ObservationGroups
+Relations
+Objects
+Subjects
+Subject
+Object
+License
+Licene
+Observations
+MultiDatastreams
+ObservedProperties
+Sensor
+Thing
+Things
+Locations
+Location
+HistoricalLocations
+FeatureOfInterest
+Party
0..*
0..1
1..*
1

Figure B.2

C

ANNEX C (INFORMATIVE) REVISION HISTORY

C ANNEX C (INFORMATIVE) REVISION HISTORY

Table C.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2022-09-21	0.1	Andreas Matheus	all	Initial version
2022-10-21	0.2	Andreas Matheus	mainly Annex A, B	Updating conformance class structure and abstract test site
2022-11-03	0.3	Andreas Matheus	section Business Logic, Annex A, B	Section added for Business Logic, Control Information and Authentication, Updating conformance class structure and abstract test site to include Business Logic, Adding Requirements and Tests for accessing personal data
2022-12-19	0.4	Andreas Matheus	STAplus Feature and Location Encoding, Annex A, Annex B, Conformance	incorporating requirements for encoding Feature and Location entities reflecting results from Sensor Things SWG session on 7 December 2022
2023-01-17	0.5	Andreas Matheus	Mainly normative sections and Annex A, B	Applied OGC NA-Policy using Metanorma annotations, Annex-B merged into Annex-A, Conformance Class definition now in section Conformance
2023-01-20	0.6	Andreas Matheus	All sections	Title adopted to include versions, updated requirements for Business Logic and CRS conformance classes, incorporated proof read from Hylke van der Schaaf
2023-02-06	0.7	Andreas Matheus	All sections	Carl Reed comments incorporated
2023-05-10	0.8	Andreas Matheus	All sections	Results from RFC incorporated, `Party` entity property `personalData` removed, Authentication class requirements and conformance tests adopted accordingly, security considerations section extended
2023-07-26	0.9	Andreas Matheus	All sections	Final edits to adopt the motion of the STA SWG from 19th July (changes in data model: Project into

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
				Campaign, Group into ObservationGroup, fixed Subject and Object associations); adding tables to improve association definitions from STA to STAplus entity types, improved wording

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] OGC, *Best Practice for using SensorThings API with Citizen Science*, (2022), https://docs.ogc.org/bp/21-068.pdf

[2] OGC, *OGC Features and Geometries JSON — Part 1: Core*, (2021), https://docs.ogc.org/DRAFTS/21-045.html[3]

[3] W3C, *Decentralized Identifiers (DIDs) v1.0*, (2022), https://www.w3.org/TR/did-core/

[4] OASIS, *OData Version 4.01. Part 1: Protocol*, (2020), http://docs.oasis-open.org/odata/odata/v4.01/odata-v4.01-part1-protocol.html.

[3]draft OGC Standard at the time of writing