OGC® DOCUMENT: 24-032R1

External identifier of this OGC® document: https://www.opengis.net/spec/sensorthings-websub/1.0

Open Geospatial Consortium

OGC SENSORTHINGS API EXTENSION: WEBSUB ASYNCHRONOUS MESSAGING STANDARD

STANDARD
Implementation

APPROVED

Version: 1.0
Submission Date: 2024-07-03
Approval Date: 2026-03-05
Publication Date: 2026-05-04
Editor: Andreas Matheus

License Agreement

Use of this document is subject to the license agreement at https://www.ogc.org/license

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

Copyright notice

Note

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

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

CONTENTS

ABSTRACT

The SensorThings API Extension: WebSub Asynchronous Messaging Standard (STA-WebSub) introduces an additional capability to the SensorThings API v1.1, allowing users to subscribe to a compliant SensorThings API endpoint using a request URL. This enables users to receive notifications when the outcome of a query changes, typically due to the arrival of new data. STA-WebSub is built upon the W3C WebSub Recommendation.

Specifically, the STA-WebSub Standard outlines how to determine if a request URL is suitable for subscription, how to initiate subscriptions using the URL, and how notifications will be delivered. This Standard mandates that a consumer implements an HTTP(S) Webhook to receive notifications from the STA-WebSub Hub(s) associated to a SensorThings API instance.

Implementing STA-WebSub enhances the versatility of SensorThings API services in creating asynchronous workflows over the HTTP protocol. The ability to establish trigger conditions and specify the structure of event data pushed to a Webhook via HTTP(S) POST allows for tailored event processing within workflows. For instance, when used alongside a STA-WebSub Subscriber that supports WebSockets, this Standard enables web applications running in modern web browsers to subscribe to and receive notifications (such as observations or other data) for seamless visualization, without the need for browser extensions. In an air quality example, a JavaScript application could alert the subscribed population about abnormal pollutant levels in real time on their mobile devices. In the weather sector, a process could be triggered that recalculates a weather forecast whenever sufficient new data is received through a subscription.

STA-WebSub provides two significant advantages. Firstly, subscribers can retrieve data from a SensorThings API service and then use the same URL to subscribe for updates. This approach eliminates the necessity for subscribers to poll the SensorThings API service. Instead, all updates tied to the specified URL are sent to the subscriber's Webhook when an update event occurs. Secondly, the MQTT protocol used by the SensorThings API does not need to be exposed to third parties for receiving updates. With STA-WebSub, the MQTT protocol can be kept internal between the SensorThings API service and the associated STA-WebSub Hub(s), enhancing security and performance management of the MQTT broker.

A compliant STA-WebSub Hub implementation must be capable to convert a subscription URL into an MQTT topic, subscribe to the MQTT broker associated with the SensorThings API service, and receive MQTT notifications to distribute them to the STA-WebSub Subscribers' Webhooks.

To fully leverage STA-WebSub's capabilities, a SensorThings API service needs to support the MQTT topic pattern to include `$filter` and `$expand` functionalities through Open Data Protocol (OData) query parameters. The `$filter` parameter supports subscriptions based on defined triggering conditions, while the use of `$select` and `$expand` allows subscribers to receive precisely the required data and structure. This feature can be described as 'receiving fit-for-purpose updates'.

To manage subscriptions according to business or governance policies, authentication and access control can be integrated into the STA-WebSub discovery process. For scalability and optimal processing of various notification types (updates), a single SensorThings API service may provide multiple STA-WebSub Hubs tailored to different types of updates or business requirements. The

assignment of a specific STA-WebSub Hub to a subscriber may depend on the subscription URL and existing business policies.

Building trust in the received updates can be achieved through the W3C WebSub's HMAC option, which offers subscribers a method to verify that the content distributed from a Hub is authentic and has not been altered.

INFO: The key work for crafting this OGC Standard was undertaken in 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.

II KEYWORDS

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

ogcdoc, OGC document, OGC Standard, API, SensorThings, W3C, WebSub, MQTT, Webhook, Publisher, Subscriber, Hub

PREFACE

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

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

IV SECURITY CONSIDERATIONS

Any event processing system is vulnerable to denial of service attacks. This susceptibility can occur when too many clients subscribe to topics that generate high-frequency updates, often referred to as 'chatty topics,' or when an excessive number of updates must be processed. For instance, subscribing to the 'chatty' MQTT topic `v1.1/Observations` poses a risk that the SensorThings API service or the STA-WebSub hub will be unable to deliver all updates to every subscribed client if the event frequency becomes excessively high. Even if the Hub can manage updates for these 'chatty' topics, subscribers may become overwhelmed with the volume of event data being sent from the Hub.

Furthermore, implementations that support ODATA query parameters are especially at risk for denial of service attacks due to the complexity involved in processing ODATA queries. An attacker could execute a denial of service attempt by subscribing through a highly complex topic URL containing an ODATA query. To mitigate such risks, implementations should carefully analyze the query parameters of a topic URL before accepting a subscription request.

SensorThings API services should deny MQTT update subscriptions that do not originate from an associated Hub. This measure ensures that an attacker is unable to bypass the Hub and directly flood the SensorThings API service with fraudulent update subscriptions. Properly separating the SensorThings API service and the Hub — such as within a fast private network — facilitates effective load distribution between various Hubs and the SensorThings API service. This adaptive scaling significantly reduces the chances of denial of service attacks, even in the case of many subscribed clients and intricate topics. Additionally, subscriptions may be denied to prevent infrastructure overload.

In any scenario, the Hub must validate a subscription with the STA-WebSub-enabled SensorThings API Service, which can be accomplished through a standard discovery request (an HTTP HEAD request containing the topic URL). To prevent fraudulent subscriptions, the Hub might require authentication for managing subscriptions.

The W3C WebSub guidelines recommend that the subscriber's callback URL should be 'not easily guessable' and that this URL should be refreshed each time a subscription is renewed. Adhering to this practice eliminates the need for other authentication mechanisms, such as API keys. An API key serves as a unique identifier that authenticates a user or application to an API instance. However, in computing environments where following this guideline is not feasible, using the `API-Key` or `X-API-Key` HTTP header can help secure a static callback endpoint URL. Additionally, when refreshing the callback URL, it is strongly advised that subscribers revise the API key value with a robust random value every time a subscription is renewed.

Leveraging an API key over HTTPS callback URLs might be preferable to using `X-Hub-Signature` when dealing with large notification data sizes. The clear advantage here is that neither the Hub nor the Subscriber is required to compute a hash over the notification data. Moreover, employing an API key over HTTPS is functionally equivalent to HMAC when distinct API keys are used for each subscription, providing authenticity and enabling the subscriber to identify the Hub. Finally, implementing HTTPS ensures content integrity.

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

1 SCOPE

1 SCOPE

This OGC SensorThings API Extension: WebSub Asynchronous Messaging Standard outlines the methods for extending the OGC SensorThings API v1.1 implementation to support asynchronous messaging over HTTP(S). This is achieved by applying the W3C WebSub Recommendation.

2 CONFORMANCE

2 CONFORMANCE

Requirements for one standardization target type are considered:

- A SensorThings API v1.1 service

Conformance with this standard SHALL be checked using all the relevant tests specified in Annex A (normative) 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® interface standard, a software implementation SHALL choose to implement:

- All of the mandatory conformance levels specified in Annex A (normative).

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

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.

Steve Liang, Tania Khalafbeigi, Hylke van der Schaaf: OGC 18-088, *OGC SensorThings API Part 1: Sensing Version 1.1*. Open Geospatial Consortium (2021). http://www.opengis.net/doc/is/sensorthings/1.1.0.

WebSub, January 23, 2018. https://www.w3.org/TR/websub

M. Nottingham: IETF RFC 5988, *Web Linking*. RFC Publisher (2010). https://www.rfc-editor.org/info/rfc5988.

4 TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

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

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

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

4.1. WebSub

> WebSub (formerly PubSubHubbub) is an open protocol for distributed publish–subscribe communication on the Internet.[1] Initially designed to extend the Atom (and RSS) protocols for data feeds, the protocol can be applied to any data type (e.g. HTML, text, pictures, audio, video) as long as it is accessible via HTTP. Its main purpose is to provide real-time notifications of changes, which improves upon the typical situation where a client periodically polls the feed server at some arbitrary interval. In this way, WebSub provides pushed HTTP notifications without requiring clients to spend resources on polling for change.
>
> — https://en.wikipedia.org/wiki/WebSub

Note 1 to entry: In October 2017, PubSubHubbub was renamed to WebSub for simplicity and clarity. As of January 2018, the WebSub protocol has been adopted by the W3C as a Recommendation.

4.2. WebSub Publisher

> an implementation that advertises a topic and hub URL on one or more resource URLs.
>
> — https://www.w3.org/TR/websub/#x3-2-1-publisher

4.3. WebSub Subscriber

> an implementation that discovers the hub and topic URL given a resource URL, subscribes to updates at the hub, and accepts content distribution requests from the hub.
>
> — https://www.w3.org/TR/websub/#x3-2-2-subscriber

4.4. WebSub Hub

> an implementation that handles subscription requests and distributes the content to subscribers when the corresponding topic URL has been updated.
>
> — https://www.w3.org/TR/websub/#x3-2-3-hub

4.5. STA-WebSub Hub

an implementation that is capable to interact with an OGC SensorThings API service to manage subscriptions and receive notifications via MQTT

4.6. Topic URL

a URL returned by the SensorThings API service as an HTTP Link header `<Link "topic URL">; rel="self"` in response to a request URL (if allowed for subscription).

4.7. MQTT Topic

a string used to subscribe to the OGC SensorThings API MQTT broker for receiving update events. The STA-WebSub Hub determines the MQTT topic from the request URL.

5 CONVENTIONS

5 CONVENTIONS

This sections 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

`https://www.opengis.net/spec/sensorthings-websub/1.0`

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

6

INTRODUCTION TO A STA-WEBSUB SYSTEM (INFORMATIVE)

6 INTRODUCTION TO A STA-WEBSUB SYSTEM (INFORMATIVE)

This section illustrates how to build a STA-WebSub System using the OGC SensorThings API Extension: WebSub Asynchronous Messaging Standard (STA-WebSub) as defined in this document. This section further outlines a complete system that extends SensorThings API into asynchronous messaging over HTTP(S).

NOTE: This section uses the natural language word “must” to highlight requirements without formal implications.

The STA-WebSub System is a specialized version of the asynchronous messaging system outlined in the W3C WebSub Recommendation. As a WebSub-based system, the STA-WebSub system consists of three components: Subscriber, Hub, and Publisher. The Publisher can be further divided into two parts: the HTTP interface of the SensorThings API service and the MQTT broker.

This STA-WebSub extension to the SensorThings API defines the requirements solely for the HTTP interface to implement the W3C WebSub Discovery protocol. Therefore, it does not affect the backward compatibility of SensorThings API version 1.1.

While the requirements for the Subscriber and Hub are also addressed in this document, they are not standardized. Appendix B offers guidance on how to implement a STA-WebSub Hub, enabling asynchronous messaging via HTTP for a SensorThings API service.

6.1. Introduction to W3C WebSub

NOTE: It is recommended to first read the W3C WebSub Recommendation.

WebSub is a W3C Recommendation that provides guidelines for implementing asynchronous processing on the web.

First, the discovery protocol enables a Subscriber to verify with a Publisher whether a self-defined topic (a resource URL) is available for subscription.

Second, the subscription protocol outlines how a Subscriber can interact with a Hub to register a topic for receiving notifications in the future.

Finally, WebSub specifies the process by which the Hub delivers updates to the Subscriber(s).

The following figure illustrates the basic functioning of the W3C WebSub Recommendation.

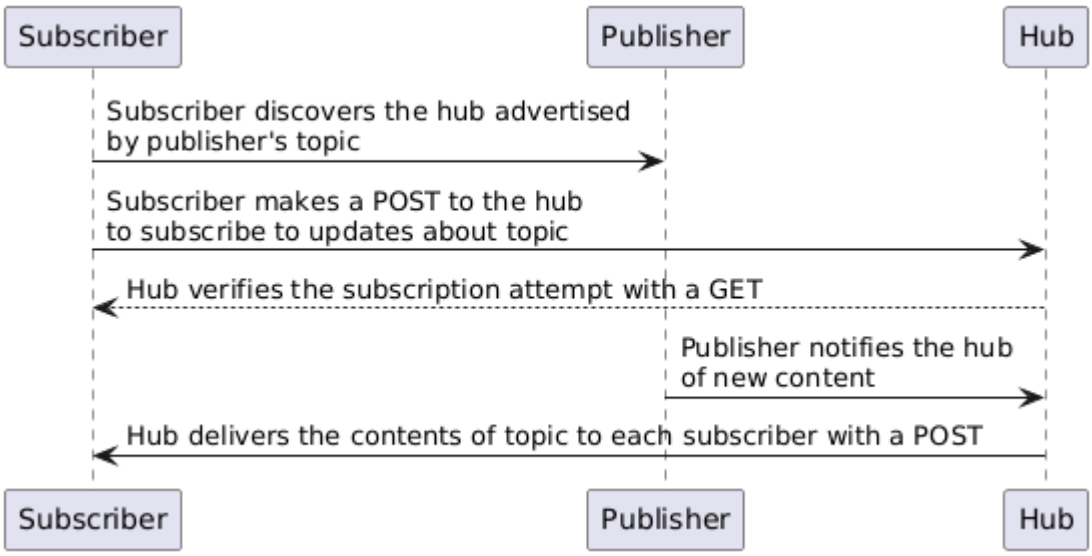


Figure 1 — W3C WebSub Flow Diagram

The notification protocol in the W3C WebSub Recommendation is unspecified (see WebSub §6 and *"Publisher notifies the hub of new content"* as illustrated above) so that basically any protocol can be used between a Publisher and a Hub. A STA-WebSub system leverages this openness to couple the Publisher (an OGC SensorThings API service) and a (STA-WebSub compliant) Hub using MQTT.

6.2. STA-WebSub Overview

An implementation of the SensorThings API Standard supports two protocols: The HTTP interface and the MQTT interface. Compliant with W3C WebSub §6, STA-WebSub specifies to use the MQTT protocol between the Hub and the Publisher. In addition the STA-WebSub extension specifies how the SensorThings API HTTP interface implementation supports the W3C WebSub discovery requirements.

A STA-WebSub system consists of four components.

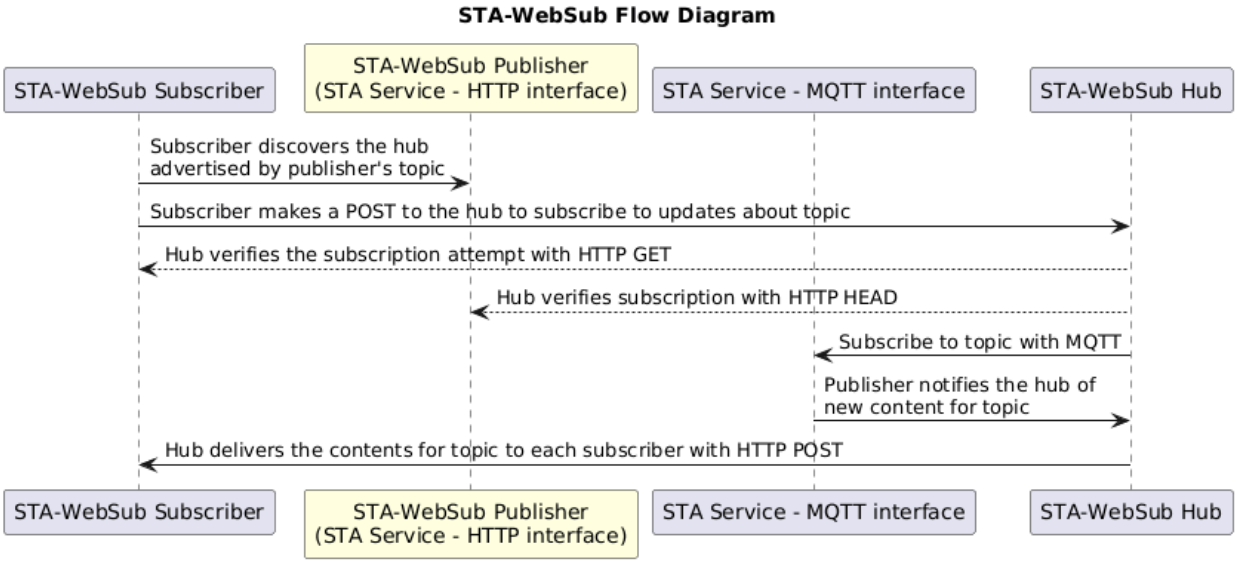
- SensorThings HTTP interface:
  - The W3C discovery protocol gets implemented here; and
  - The STA-WebSub error handling gets implemented here.
- SensorThings MQTT broker:
  - The Hub subscribes (and unsubscribes) for updates on MQTT topics; and
  - The broker sends MQTT notifications to the Hub.

- Hub:
  - The subscription functionality must be extended to transform a W3C topic URL into a SensorThings MQTT topic and the Hub must use the MQTT protocol to subscribe/unsubscribe for topics; and
  - The notification of distribution events takes place via MQTT. A STA-WebSub Hub must therefore listen to MQTT events received from the associated MQTT broker and distribute the event content to all subscribers using HTTP(S) POST.

- Subscriber:
  - The Subscriber discovers subscription topics by sending HTTP HEAD and GET request URLs to the HTTP interface of the SensorThings API service;
  - A URL that can be used for subscription with the Hub (via `hub.topic`) can be obtained from the HTTP Link header `rel="self"` included in the discovery response; and
  - A subscriber may use an `api-key` or `x-api-key` parameter for subscribing to the Hub.

The STA-WebSub — SensorThings API extension WebSub — defines additional capabilities in the context of SensorThings API that allow the propagation of updates via HTTP(S) using callback URLs to WebHooks as defined in the W3C WebSub Recommendation. The notification update is based on an event for a MQTT topic as defined in the SensorThings API Standard.

To make STA-WebSub work, the Subscriber, Publisher and Hub need to implement certain functionalities. In order to understand the requirements better, the following high-level flow diagram illustrates the additions for STA-WebSub.



NOTE: As illustrated in the STA-WebSub flow diagram above, this standard has implications for the subscriber, the publisher's implementations, and the Hub. However, only the publisher's

section (highlighted in the yellow box) is considered normative in this document, as the SensorThings API service is the target of standardization.

To ensure a functional solution, Appendix B offers guidance on the implementation-specific aspects necessary to adapt a W3C Hub and implement a STA-WebSub Hub.

Figure 2 — STA-WebSub Flow Diagram

The W3C WebSub specification does not define specific topics. Instead, a WebSub Subscriber can determine whether a resource URL is available for subscription. This flexibility is crucial for the STA-WebSub to enable subscribers to define:

- The actual trigger for an update event, and
- The data structure for the information delivered with an update event using ODATA query parameters.

Any implementation of the STA-WebSub extension must support the MQTT topic pattern as outlined in the SensorThings API v1.1 Standard, enhanced with ODATA queries. A SensorThings API service that accommodates ODATA options with MQTT topics allows subscribers to utilize `$filter` for specifying event triggers, as well as `$select` and `$expand` to define the content data structure.

The capacity to create notification conditions combined with the ability to specify the precise data structure required for a notification is a powerful feature. This capability is essential for executing workflows tailored to a subscriber's Webhook.

NOTE: The support for `$select` is mandatory for a SensorThings API service implementation. However, the support for `$filter` and `$expand` is optional.

6.3. Introduction to Discovery, Subscription and Notification

Certain functional requirements regarding discovery, subscription and notification need to be defined for extending the W3C WebSub protocol to support STA-WebSub.

6.3.1. Discovery

A STA-WebSub compliant SensorThings API HTTP interface (also known as a Publisher) supports W3C WebSub discovery by adding `Link` headers with `rel="hub"` and `rel="self"` to the HTTP response. W3C WebSub additionally requires that these `Link` headers be included either as HTTP response headers or inline in an XHTML encoded response.

Since the typical response format for a SensorThings API implementation is either JSON or GeoJSON, rather than XHTML, the STA-WebSub extension mandates that the `Link` headers be

returned as HTTP response headers. Furthermore, W3C WebSub expects these discovery links to be available in response to both HTTP HEAD and GET requests.

To comply with this requirement for discovery, the STA-WebSub extension stipulates that a SensorThings API HTTP interface must support the HTTP HEAD method in addition to the already existing HTTP GET method.

The following sequence diagram illustrates the discovery for the URL `http://localhost/sta/v1.1/Observations`

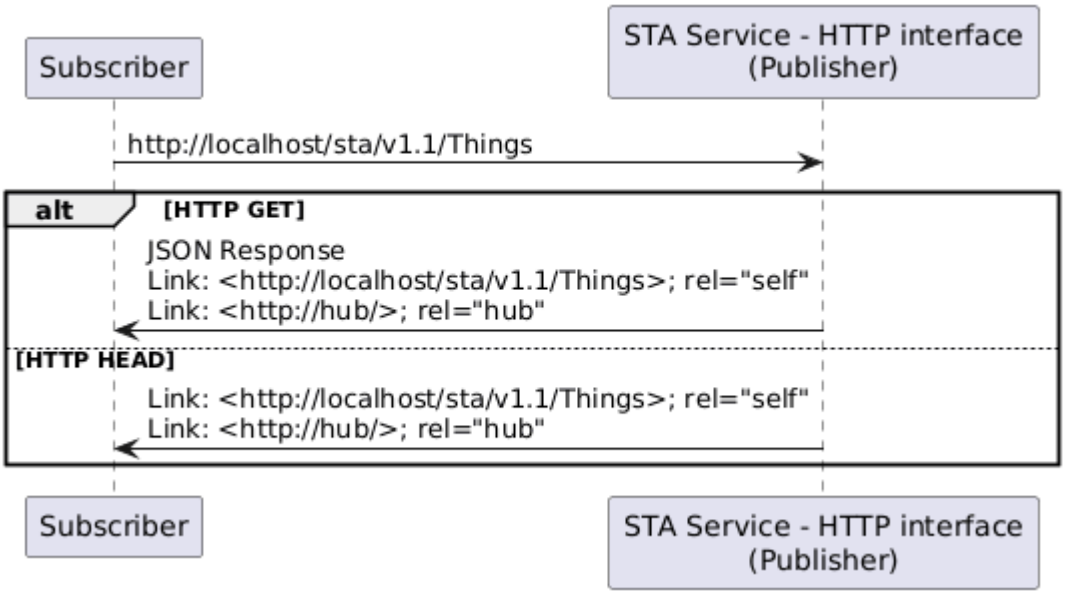


Figure 3 — Discovery with URL for supported subscription

For this URL, the implementation returns the `<Link>; rel="self"` header.

6.3.2. Discovery "Error" Handling

A STA-WebSub hub may receive a subscription request via the `hub.topic`, which translates into an MQTT topic. However, this topic may not be accepted by the SensorThings API MQTT broker. The accepted topic URLs are deployment-specific. For instance, subscribing to `/Observations` might generate an excessive load, resulting in it being disallowed. Additionally, using unsupported or disallowed OData options, such as `$expand` or `$filter`, can cause the absence of a `Link rel="self"` header in the response. The lack of this header indicates that a subscription for that particular topic URL is not possible. This behavior is perfectly in line with the W3C WebSub Recommendation. Nonetheless, any subscriber, whether a user or a service/process, may be puzzled as to why the discovery response does not include the `Link rel="self"` header.

It is important to note that the absence of a `Link rel="self"` header is not considered an error. Therefore, using HTTP 4xx status codes in this case would be inappropriate. Furthermore, the W3C WebSub specification does not provide guidelines for error handling in these scenarios.

There should be guidance available to clarify why the `Link rel="self"` header is missing, which would support users or developers implementing a STA-WebSub Subscriber.

To address this issue, this STA-WebSub Standard introduces the `Link rel="help"` header. The URL associated with this relationship should direct users to a static help page that explains why a subscription to the specified topic URL is not permitted.

The following sequence diagram illustrates the discovery of a URL that is unsupported for subscription: `http://localhost/sta/v1.1/Observations`.

NOTE: It is assumed that the topic `v1.1/Observation` is blacklisted.

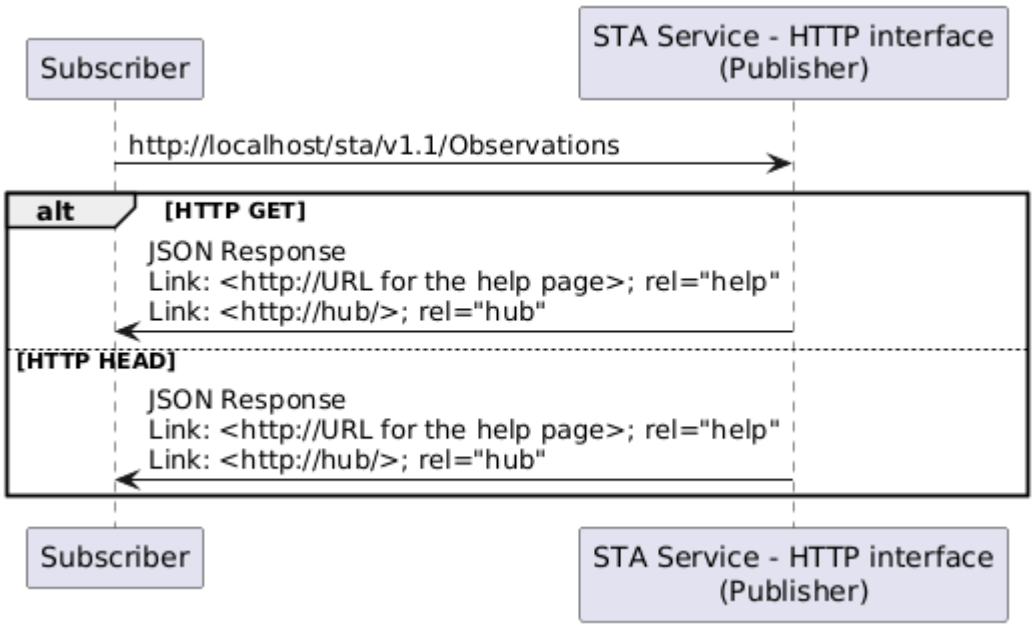


Figure 4 — Discovery with URL not supported for subscription due to topic restriction

To acknowledge that the topic URL cannot be used for subscription, the implementation does not return the `<Link>; rel="self"`. The `<Link>; rel="help"` URL URL explains the error: The topic URL is disallowed for subscription because it is using disallowed ODATA option (ODATA option `$expand` is blacklisted): `http://localhost/sta/v1.1/Datastreams(4711)/Observations?$expand=FeatureOfInterest`

A STA-WebSub compliant implementation does not return the `<Link>; rel="self"` but the `<Link>; rel="help"` header for the URL above.

6.3.3. Subscriptions

A STA-WebSub Hub implementation must be capable of transforming the W3C WebSub `hub.topic` expressed as an HTTP(S) URL into an MQTT topic pattern that is accepted by the MQTT broker associated with the SensorThings API service. Given that the SensorThings API service and the Hub closely interact, this transformation should not be overly complicated.

To prevent subscriptions to unsupported MQTT topics, the Hub must utilize the discovery protocol and deny any requests if the discovery response does not include the `Link rel= "self"` header.

If the Hub receives an unsubscribe request from a subscriber, it must confirm the intent with the subscriber and proceed to unsubscribe from the MQTT topic linked to the associated SensorThings API service.

6.3.4. Notifications

Once the Hub subscribes to an MQTT topic, it awaits notifications from the MQTT broker associated with the SensorThings API service.

When a notification event occurs (consisting of both the topic and content), the Hub delivers the content to the callback URLs that have subscribed (the subscribers' Webhooks) using an HTTP POST request.

6.4. Root Page

The support for STA-WebSub is advertised on the SensorThings API root page. The SenorThings API root page is the response to an HTTP GET request issued to the service root URI as defined in OGC 18-088, §9. A compliant implementation and deployment of a SensorThings API service supporting STA-WebSub lists the following conformance classes under the `serverSettings/conformance`:

- https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery

The blacklisting of ODATA options and topics is also advertised.

Example 1: A STA-WebSub compliant SensorThings API service that has no ODATA options restrictions and not restrictions of topics

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": []
        }
    }
}
```

Example 2: A STA-WebSub compliant SensorThings API service that has ODATA options `$expand` and `$filter` restrictions and no restrictions of topics

```
{
```

```
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": ["$expand", "$filter"]
        }
    }
}
```

Example 3: A STA-WebSub compliant SensorThings API service that has no ODATA options restrictions but restriction of topic `v1.1/Observations`

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": ["v1.1/Observations"],
            "odata_denied": []
        }
    }
}
```

6.5. Benefits of STA-WebSub

Compared to PubSub, where subscriptions are defined by the Publisher, one of the biggest benefits for STA-WebSub Subscribers is that they can determine whether a self-defined topic is supported for receiving notifications and can receive those notifications via HTTP(S) POST. For providers, there is the advantage of being able to control topic discovery, which allows them to manage the use of the MQTT interface for sending notifications and to scale through STA-WebSub Hub redundancy.

The OGC SensorThings API Standard allows MQTT clients to subscribe to update events, meaning that these events are delivered from the MQTT broker to all subscribed clients. The STA-WebSub extension, as defined in this OGC Standard, facilitates the distribution of MQTT events via HTTP(S). Consequently, the MQTT broker does not directly send update events to subscribers; rather, it is the Hub that handles this distribution using HTTP(S). This separation of duties offers important improvements in terms of usability, security, and scalability.

By using the STA-WebSub extension, the MQTT protocol remains internal between the MQTT broker of the SensorThings API service and the associated Hub(s). This allows for control over the origin of MQTT subscriptions at the associated STA-WebSub Hub. Additionally, employing discovery policies enables the implementation of flexible and fine-grained access control regarding subscriptions.

The internal use of the MQTT protocol between the Hub and the SensorThings API Service simplifies subscriber interactions with well-known infrastructure patterns, such as HTTP Webhooks, effectively utilizing a W3C WebSub compliant HTTP(S) endpoint that listens for GET and POST requests.

The separation of responsibilities for sending update content to subscribers, as managed by both the SensorThings API and the Hub, enhances scalability. The SensorThings API service is responsible for delivering topic updates to associated Hub(s) via MQTT, after which the Hub(s) can process the MQTT messages and distribute the content to subscribers using established cloud-scaling software stacks.

The capability for subscribers to define notification conditions (e.g., using `$filter`) and the structure of the notification data (e.g., using `$select` and `$expand`) enhances usability compared to predefined MQTT topics. The extent of flexibility available to a subscriber is determined by the discovery functionality that reflects support for `$select` and `$expand`.

The ability to handle subscriptions and publications through HTTP, implemented by the Webhook, enables workflow execution and asynchronous messaging to purely HTML5/ JavaScript web client, where updates are received via WebSocket without requiring any extensions to be set up in the web browsers.

7

SENSORTHINGS API – WEBSUB EXTENSION (NORMATIVE)

7 SENSORTHINGS API — WEBSUB EXTENSION (NORMATIVE)

As outlined in Clause 6, a system leveraging the SensorThings API Extension: WebSub Asynchronous Messaging (STA-WebSub) is a specialization of the asynchronous messaging system described in the W3C WebSub Recommendation. As a W3C WebSub system, the STA-WebSub system is comprised of three parts (Subscriber, Hub, Publisher). In the context of this OGC SensorThings API Extension: WebSub Asynchronous Messaging Standard, the Publisher is the SensorThings API service which can be split into two sub-components: the HTTP interface and the MQTT broker.

This STA-WebSub extension defines one requirements class:

- Discovery Requirements Class (mandatory)

NOTE:Appendix B provides guidance for implementing a STA-WebSub Hub.

7.1. Discovery Requirements Class (mandatory)

The `Discovery` requirements class is implemented at the SensorThings API service — HTTP interface to support the W3C WebSub discovery protocol.

REQUIREMENTS CLASS 1: REQUIREMENTS CLASS 'DISCOVERY'	
IDENTIFIER	`https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
TARGET TYPE	Web API
NORMATIVE STATEMENTS	Requirement 1: `/req/req-class-discovery/req-landing-page-discovery` Requirement 2: `/req/req-class-discovery/req-landing-page-topics` Requirement 3: `/req/req-class-discovery/req-landing-page-odata` Requirement 4: `/req/req-class-discovery/req-http-methods` Requirement 5: `/req/req-class-discovery/req-link-hub` Requirement 6: `/req/req-class-discovery/req-link-self` Requirement 7: `/req/req-class-discovery/req-link-help` Requirement 8: `/req/req-class-discovery/req-odata-support` Requirement 9: `/req/req-class-discovery/req-odata-discovery` Requirement 10: `/req/req-class-discovery/req-odata-blacklisting` Requirement 11: `/req/req-class-discovery/req-topics-discovery` Requirement 12: `/req/req-class-discovery/req-topics-blacklisting`

7.1.1. Root Page Discovery Requirements Class (mandatory)

According to OGC 18-088 a compliant implementation returns a root page.

REQUIREMENT 1	
IDENTIFIER	`/req/req-class-discovery/req-landing-page-discovery`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL return a SensorThings API v1.1 compliant root page.
B	The root page SHALL include the value `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` in the `conformance` array.

REQUIREMENT 2	
IDENTIFIER	`/req/req-class-discovery/req-landing-page-topics`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL include the key `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` into the `serverSettings` of the root page.
B	The key `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` SHALL contain the key `topics_denied`.
C	The key `topics_denied` SHALL include either an empty JSON array or a JSON array that contains all denied topics.

REQUIREMENT 3	
IDENTIFIER	`/req/req-class-discovery/req-landing-page-odata`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL include the key `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` into the `serverSettings` of the root page.
B	The key `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` SHALL contain the key `odata_denied`.

REQUIREMENT 3	
C	The key `odata_denied` SHALL include either an empty JSON array or a JSON array that contains all denied ODATA options.

NOTE: The root-page requirements group JSON structure obligations together for ease of reference.

7.1.2. Requirement HTTP Methods

According to W3C WebSub Recommendation, discovery can be achieved via HTTP GET and HEAD methods.

REQUIREMENT 4	
IDENTIFIER	`/req/req-class-discovery/req-http-methods`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL support discovery via HTTP GET method.
B	An implementation SHALL support discovery via HTTP HEAD method.

7.1.3. Requirement Link Header 'hub'

According to W3C WebSub Recommendation, discovery always returns `<Link>; rel="hub"`.

REQUIREMENT 5	
IDENTIFIER	`/req/req-class-discovery/req-link-hub`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL return the `<Link>; rel="hub"` HTTP header to indicate support for WebSub.

7.1.4. Requirement Link Header 'self'

According to W3C WebSub Recommendation, discovery returns `<Link>; rel="self"` for a URL that is suitable for subscription.

REQUIREMENT 6	
IDENTIFIER	`/req/req-class-discovery/req-link-self`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL return the `<Link>; rel="self"` HTTP header to indicate the ability for subscription.
B	An implementation SHALL not return the `<Link>; rel="self"` HTTP header if the request URL cannot be used for subscription. In this case, the implementation SHALL return the `<Link>; rel="help"`.
C	An implementation SHALL determine if a request URL is allowed for subscription by first converting the URL into an MQTT topic and test whether the topic is included in the `topics_denied`. If so, the implementation SHALL not return the `<Link>; rel="self"` but the `<Link>; rel="help"` HTTP header.
D	An implementation SHALL isolate the query string from the request URL and check if any of the ODATA options is listed in the `odata_denied`. If so, the implementation SHALL not return the `<Link>; rel="self"` but the `<Link>; rel="help"` HTTP header.
E	An implementation SHALL determine other sources to determine if the request URL can be used for subscription. If the request URL cannot be used for subscription, the implementation SHALL not return the `<Link>; rel="self"` but the `<Link>; rel="help"` HTTP header.

7.1.5. Requirement Link Header ‘help’

Returning the `<Link>; rel="help"` header (specific to STA-WebSub) indicates the cause why the requested URL is not suitable for subscription.

REQUIREMENT 7	
IDENTIFIER	`/req/req-class-discovery/req-link-help`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL provide a URL for obtaining help (why the request URL cannot be used for subscription) when returning the `<Link>; rel="help"` header.
B	An implementation SHALL use the `<Link>; rel="self"` and`<Link>; rel=”help”` HTTP header mutually exclusive.

7.1.6. Requirement ODATA query options support

REQUIREMENT 8	
IDENTIFIER	/req/req-class-discovery/req-odata-support
INCLUDED IN	Requirements class 1: https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery
A	An implementation SHALL allow the discovery such that the associated MQTT topic is compliant to OGC SensorThings API v1.1 requirement regarding updates via MQTT [section 14.2 — https://docs.ogc.org/is/18-088/18-088.html#req-receive-updates-via-mqtt-receive-updates] with the following extensions: • SERVICE_VERSION/RESOURCE_PATH/COLLECTION_NAME as defined in 14.2.1 can be extended with a ? followed by any valid combination of ODATA options — e.g. v1.1/Datastreams(1)/Observations?$filter=result gt 30 • SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY as defined in 14.2.2 can be extended with a ? followed by any valid combination of ODATA options — e.g. v1.1/Observations?$select=result • SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY/PROPERTY_NAME as defined in 14.2.3

7.1.7. Requirement ODATA support advertisement

REQUIREMENT 9	
IDENTIFIER	/req/req-class-discovery/req-odata-discovery
INCLUDED IN	Requirements class 1: https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery
A	An implementation SHALL advertise the support for STA-WebSub by publishing the string https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery in the serverSettings section of the root page.

7.1.8. Requirement ODATA Blacklisting

A request URL may contain query parameters (after the ?) presenting ODATA options. An implementation may deny discovery (and thereby later subscription) for certain ODATA options like $expand or $filter.

REQUIREMENT 10	
IDENTIFIER	/req/req-class-discovery/req-odata-blacklisting
INCLUDED IN	Requirements class 1: https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery

REQUIREMENT 10	
A	An implementation SHALL advertise the existence of denied ODATA options by adding the following key to the root page: `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` that includes the key `odata_denied`. This key SHALL contain a JSON array for all denied ODATA options.
B	In case no ODATA options are denied, an implementation SHALL advertise the non-existence of denied ODATA options by adding the following key to the root page: `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` that includes the key `odata_denied`. This key SHALL contain an empty JSON array.

Example 1: Example blacklisting structure for ODATA options

```
{
    "serverSettings": {
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "odata_denied": <JSON Array of denied ODATA options>
        }
    }
}
```

Example 2: For example, the following root page snippet indicates restrictions for `$expand`, `$skip`, `$top` and `$filter`

```
{
    "serverSettings": {
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "odata_denied": ["$expand", "$skip", "$top", "$filter"]
        }
    }
}
```

Example 3: The following root page snippet indicates **no** restrictions for ODATA options

```
{
    "serverSettings": {
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "odata_denied": []
        }
    }
}
```

7.1.9. Requirement Topics Blacklisting

The support for STA-WebSub can be determined from the root page as part of the `Discovery` conformance class.

REQUIREMENT 11

IDENTIFIER	`/req/req-class-discovery/req-topics-discovery`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	An implementation SHALL advertise the support for MQTT topic blacklisting by publishing the string `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` in the `serverSettings` section of the root page that includes the key `topics_denied`.

Example 1: Example blacklisting structure for topics

```
{
    "serverSettings": {
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": <JSON Array of denied topics>
        }
    }
}
```

REQUIREMENT 12

IDENTIFIER	`/req/req-class-discovery/req-topics-blacklisting`
INCLUDED IN	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
A	A STA topic starts with `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY` as defined in 14.2.2.
B	An implementation SHALL advertise the existence of denied topics by adding the following key to the root page: `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` that includes the key `topics_denied`. This key SHALL contain a JSON array listing all denied topics.
C	In case no topics are denied, an implementation SHALL advertise the non-existence of denied topics by adding the following key to the root page: `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` that includes the key `topics_denied`. This key SHALL contain an empty JSON array.

Example 2: The following root page snippet indicates deny for the topics `v1.1/Observations` and `v1.1/Datastreams('very chatty')/Observations`

```
{
    "serverSettings": {
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": ["v1.1/Observations", "v1.1/Datastreams('very
chatty')/Observations"]
        }
    }
}
```

Example 3: The following root page snippet indicates **no** deny for topics:

```
{
    "serverSettings": {
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": []
        }
    }
}
```

8

MEDIA TYPES FOR ANY DATA ENCODING(S)

8 MEDIA TYPES FOR ANY DATA ENCODING(S)

This OGC Standard uses Link Header as defined in IETF RFC 5988 and W3C WebSub Recommendation.

In particular, the following HTTP Link headers are used:

- `rel="hub"` as defined in https://www.w3.org/TR/websub/#x4-discovery
- `rel="self"` as defined in https://www.w3.org/TR/websub/#x4-discovery
- `rel="help"` as defined in https://www.iana.org/assignments/link-relations/link-relations.xhtml

A

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

ANNEX A (NORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

For compliance, the mandatory conformance tests must be implemented.

A.1. Conformance Class Discovery (mandatory)

CONFORMANCE CLASS A.1	
SUBJECT	Requirements Class "Discovery"
REQUIREMENTS CLASS	Requirements class 1: `https://www.opengis.net/spec/sensorthings-websub/1.0/req/discovery`
TARGET TYPE	Web API
LABEL	https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery
CONFORMANCE TESTS	Abstract test A.1: `/conf/discovery/api-sta-websub-landing-page-discovery` Abstract test A.2: `/conf/discovery/api-sta-websub-landing-page-topics` Abstract test A.3: `/conf/discovery/api-sta-websub-landing-page-odata` Abstract test A.4: `/conf/discovery/api-sta-websub-http-get` Abstract test A.5: `/conf/discovery/api-sta-websub-http-head` Abstract test A.6: `/conf/discovery/api-sta-websub-link-header-hub` Abstract test A.7: `/conf/discovery/api-sta-websub-link-header-self` Abstract test A.8: `/conf/discovery/api-sta-websub-link-header-help` Abstract test A.9: `/conf/discovery/api-sta-websub-odata-support` Abstract test A.10: `/conf/discovery/api-sta-websub-odata-discovery` Abstract test A.11: `/conf/discovery/api-sta-websub-odata-blacklisting` Abstract test A.12: `/conf/discovery/api-sta-websub-odata-no-blacklisting` Abstract test A.13: `/conf/discovery/api-sta-websub-topics-discovery` Abstract test A.14: `/conf/discovery/api-sta-websub-topics-blacklisting` Abstract test A.15: `/conf/discovery/api-sta-websub-topics-no-blacklisting`

A.1.1. Root Page

ABSTRACT TEST A.1	
IDENTIFIER	`/conf/discovery/api-sta-websub-landing-page-discovery`
SUBJECT	/req/req-class-discovery/req-landing-page-discovery
REQUIREMENT	Requirement 1: `/req/req-class-discovery/req-landing-page-discovery`
LABEL	/conf/discovery/api-sta-websub-landing-page-discovery
TEST PURPOSE	Validate that the implemented STA-WebSub Conformance Class `Discovery` is listed on the Root Page.
TEST METHOD	1. Construct a URL to the Root Page. 2. Issue a HTTP GET request on that URL. 3. Validate that the response contains a root page compliant with SensorThings API v1.1. 4. Validate the contents of the returned document to include the implemented Conformance Class(es): `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery`

A.1.2. Example

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        }
    }
}
```

Listing A.1

ABSTRACT TEST A.2	
IDENTIFIER	`/conf/discovery/api-sta-websub-landing-page-topics`
SUBJECT	/req/req-class-discovery/req-landing-page-topics
REQUIREMENT	Requirement 2: `/req/req-class-discovery/req-landing-page-topics`

ABSTRACT TEST A.2	
LABEL	/conf/discovery/api-sta-websub-landing-page-topics
TEST PURPOSE	Validate that the denied root topics are advertised on the Root Page.
TEST METHOD	1. Construct a URL to the Root Page. 2. Issue a HTTP HEAD and GET request on that URL. 3. Validate the contents of the returned document to include the implemented Conformance Class(es): `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` 4. Validate the contents of the returned document to include the JSON object `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` including the key `topics_denied`. 5. Validate that the value of the key `topics_denied` is a JSON array that it is either empty ("[]") or contains a list of denied root topics. 6. Validate that each value of the key `topics_denied` is compliant to `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY`. **NOTE:** A root topic starts with `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY` as defined in 14.2.2

A.1.3. Example Root Page snippet advertising no denied root topics

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": []
        }
    }
}
```

Listing A.2

A.1.4. Example Root Page snippet advertising `v1.1/Observations` as a denied root topic

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
```

```
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": ["v1.1/Observations"]
        }
    }
}
```

Listing A.3

ABSTRACT TEST A.3	
IDENTIFIER	`/conf/discovery/api-sta-websub-landing-page-odata`
SUBJECT	/req/req-class-discovery/req-landing-page-odata
REQUIREMENT	Requirement 3: `/req/req-class-discovery/req-landing-page-odata`
LABEL	/conf/discovery/api-sta-websub-landing-page-odata
TEST PURPOSE	Validate that the denied ODATA commands are advertised on the Root Page.
TEST METHOD	1. Construct a URL to the Root Page. 2. Issue a HTTP HEAD and GET request on that URL. 3. Validate the contents of the returned document to include the implemented Conformance Class(es): `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` 4. Validate the contents of the returned document to include the JSON object `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` including the key `odata_denied`. 5. Validate that the value of the key `odata_denied` is a JSON array that it is either empty (“[]”) or contains a list of denied ODATA options.

A.1.5. Example Root Page snippet advertising no denied ODATA options

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "odata_denied": []
        }
    }
}
```

Listing A.4

A.1.6. Example Root Page snippet advertising `$expand` as a denied ODATA options

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "odata_denied": ["$expand"]
        }
    }
}
```

Listing A.5

A.1.7. HTTP Methods

ABSTRACT TEST A.4	
IDENTIFIER	`/conf/discovery/api-sta-websub-http-get`
SUBJECT	/req/req-class-discovery/req-http-methods
REQUIREMENTS	Requirement 4: `/req/req-class-discovery/req-http-methods` Requirement 5: `/req/req-class-discovery/req-link-hub`
LABEL	/conf/discovery/api-sta-websub-http-get
TEST PURPOSE	Validate that the STA-WebSub discovery is supported via HTTP GET.
TEST METHOD	1. Construct a topic URL to be used for subscription. 2. Issue a HTTP GET request on that topic URL. 3. Validate that the request method was accepted. 4. Validate that the response headers include the Link header `hub`.

ABSTRACT TEST A.5	
IDENTIFIER	`/conf/discovery/api-sta-websub-http-head`
SUBJECT	/req/req-class-discovery/req-http-methods

ABSTRACT TEST A.5	
REQUIREMENTS	Requirement 4: `/req/req-class-discovery/req-http-methods` Requirement 5: `/req/req-class-discovery/req-link-hub`
LABEL	/conf/discovery/api-sta-websub-http-head
TEST PURPOSE	Validate that the STA-WebSub discovery is supported via HTTP HEAD.
TEST METHOD	1. Construct a topic URL to be used for subscription. 2. Issue a HTTP HEAD request on that topic URL. 3. Validate that the request method was accepted. 4. Validate that the response headers include the Link header `hub`.

A.1.8. Link Headers

ABSTRACT TEST A.6	
IDENTIFIER	`/conf/discovery/api-sta-websub-link-header-hub`
SUBJECT	/req/req-class-discovery/req-link-hub
REQUIREMENT	Requirement 5: `/req/req-class-discovery/req-link-hub`
LABEL	/conf/discovery/api-sta-websub-link-header-hub
TEST PURPOSE	Validate that the WebSub discovery returns the link headers `hub` for any topic URL.
TEST METHOD	1. Construct a topic URL. 2. Issue a HTTP HEAD request on that topic URL. 3. Validate that the response includes the Link header `hub`. 4. Issue a HTTP GET request on that topic URL. 5. Validate that the response includes the Link header `hub`.

ABSTRACT TEST A.7	
IDENTIFIER	`/conf/discovery/api-sta-websub-link-header-self`
SUBJECT	/req/req-class-discovery/req-link-self

ABSTRACT TEST A.7

REQUIREMENTS	Requirement 6: `/req/req-class-discovery/req-link-self` Requirement 5: `/req/req-class-discovery/req-link-hub`
LABEL	/conf/discovery/api-sta-websub-link-header-self
TEST PURPOSE	Validate that the WebSub discovery returns the link headers `hub` and `self` for a topic URL that is **allowed** for subscription.
TEST METHOD	1. Construct topic URLs that are **allowed** for subscription. Construct a topic URL following topic pattern `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY/PROPERTY_NAME` where the topic is not included in `topic_denied`. Construct a topic URL following the topic patterns `SERVICE_VERSION/RESOURCE_PATH/COLLECTION_NAME` and `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY` excluding ODATA options or including ODATA options that are not listed in `odata_denied`. 2. Issue a HTTP HEAD and GET request on each topic URL. 3. Validate that the response includes the Link header `hub`. 4. Validate that the response includes the Link header `self`.

ABSTRACT TEST A.8

IDENTIFIER	`/conf/discovery/api-sta-websub-link-header-help`
SUBJECT	/req/req-class-discovery/req-link-help
REQUIREMENTS	Requirement 7: `/req/req-class-discovery/req-link-help` Requirement 6: `/req/req-class-discovery/req-link-self` Requirement 5: `/req/req-class-discovery/req-link-hub`
LABEL	/conf/discovery/api-sta-websub-link-header-help
TEST PURPOSE	Validate that the WebSub discovery returns the link headers `hub` and `help` for a topic URL **not allowed** for subscription.
TEST METHOD	1. Construct a topic URL that is **not allowed** for subscription based on `topics_denied`. Construct a topic URL that is **not allowed** for subscription based on `odata_denied`. Construct a topic URL that is **not allowed** for subscription if reasons exist to deny subscription other than based on `topics_denied` and `odata_denied`. 2. Issue a HTTP HEAD and GET request for each topic URL. 3. Validate that the response includes the Link header `hub`. 4. Validate that the response includes the Link header `help`. 5. Validate that the URL from the Link header `help` is resolvable. 6. Validate that the content from the URL from the Link header `help` contains useful information describing the reason why the Link `self` is not present. 7. Validate that the response does **not** include the Link header `self`.

A.1.9. Example for topics and ODATA restrictions

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": ["v1.1/Observations"],
            "odata_denied": ["$expand"]
        }
    }
}
```

Listing A.6

A.1.10. Example HTTP request and response for a URL including the denied topic `v1.1/Observations`

```
GET /sta/v1.1/Observations HTTP/1.1
Host: localhost

HTTP/1.1 200 Ok
Content-type: application/json
Link: <http://hub.localhost>; rel="hub"
Link: <http://localhost/sta/v1.1/help#topic_denied>; rel="help"
```

Listing A.7

A.1.11. Example HTTP request and response for a URL including the denied ODATA option `$expand`

```
GET /sta/v1.1/Things?$expand=Locations HTTP/1.1
Host: localhost

HTTP/1.1 200 Ok
Content-type: application/json
Link: <http://hub.localhost>; rel="hub"
Link: <http://localhost/sta/v1.1/help#odata_option_denied>; rel="help"
```

Listing A.8

A.1.12. Tests for ODATA Support

ABSTRACT TEST A.9	
IDENTIFIER	/conf/discovery/api-sta-websub-odata-support

ABSTRACT TEST A.9	
SUBJECT	/req/req-class-discovery/req-odata-support
REQUIREMENT	Requirement 8: `/req/req-class-discovery/req-odata-support`
LABEL	/conf/discovery/api-sta-websub-odata-support
TEST PURPOSE	Validate that the SensorThings implementation supports the use of ODATA options.
TEST METHOD	1. Construct SensorThings URLs for each valid and implemented ODATA option(s) as advertised on the root page. Construct a topic URL following pattern `SERVICE_VERSION/RESOURCE_PATH/COLLECTION_NAME` for each ODATA option advertised on the root page. Construct a topic URL following pattern `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY` for each ODATA option advertised on the root page. Any constructed topic URL must not follow pattern `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY/PROPERTY_NAME`. 2. Issue a HTTP GET request for each topic URL. 3. Validate that each response for a topic URL following the topic pattern `SERVICE_VERSION/RESOURCE_PATH/COLLECTION_NAME` or `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY` is compliant with SensorThings API v1.1.

A.1.13. Tests for ODATA Support Advertisement

ABSTRACT TEST A.10	
IDENTIFIER	`/conf/discovery/api-sta-websub-odata-discovery`
SUBJECT	/req/req-class-discovery/req-odata-discovery
REQUIREMENT	Requirement 9: `/req/req-class-discovery/req-odata-discovery`
LABEL	/conf/discovery/api-sta-websub-odata-discovery
TEST PURPOSE	Validate that the disallowed ODATA options are advertised on the root page.
TEST METHOD	1. Construct a URL to the root page. 2. Issue a HTTP HEAD and GET request on that URL. 3. Validate the contents of the returned document to include the implemented Conformance Class(es) in the `conformance` section of the root page: `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` 4. Validate the contents of the returned document to include the JSON object `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` in the `serverSettings` including the key `odata_denied`.

ABSTRACT TEST A.10

5. Validate that the value of the key `odata_denied` is a JSON array that is either empty (“[]”) or contains a list of denied ODATA options.

A.1.14. Example root page snippet advertising no denied ODATA options

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": []
        }
    }
}
```

Listing A.9

A.1.15. Example root page snippet advertising `$expand` as a denied ODATA option

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": ["$expand"]
        }
    }
}
```

Listing A.10

A.1.16. Tests for ODATA Options Blacklisting

ABSTRACT TEST A.11

IDENTIFIER	`/conf/discovery/api-sta-websub-odata-blacklisting`
SUBJECT	/req/req-class-discovery/req-odata-blacklisting

ABSTRACT TEST A.11	
REQUIREMENT	Requirement 10: `/req/req-class-discovery/req-odata-blacklisting`
LABEL	/conf/discovery/api-sta-websub-odata-blacklisting
TEST PURPOSE	Validate that the disallowed ODATA options do not return the HTTP link header `rel="self"`.
TEST METHOD	1. Construct multiple SensorThings URLs each including a supported ODATA option that is listed under the `odata_denied`. 2. Issue HTTP HEAD and GET requests on each URL. 3. Validate the contents of the returned HTTP header to not include the link header `rel="self"` but the link header `rel="help"`. 4. Validate the contents of the returned HTTP header to include the link header `rel="hub"`.

A.1.17. Example root page with denied ODATA option `$expand`

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": ["$expand"]
        }
    }
}
```

Listing A.11

A.1.18. Example HTTP response for a URL including the denied ODATA option `$expand`

```
GET /sta/v1.1/Things?$expand=Locations HTTP/1.1
Host: localhost

HTTP/1.1 200 Ok
Content-type: application/json
Link: <http://hub.localhost>; rel="hub"
Link: <http://localhost/sta/v1.1/help#odata_option_denied>; rel="help"
```

Listing A.12

A.1.19. Tests for no Blacklisting of ODATA Options

ABSTRACT TEST A.12	
IDENTIFIER	`/conf/discovery/api-sta-websub-odata-no-blacklisting`
SUBJECT	/req/req-class-discovery/req-odata-blacklisting
REQUIREMENT	Requirement 10: `/req/req-class-discovery/req-odata-blacklisting`
LABEL	/conf/discovery/api-sta-websub-odata-no-blacklisting
TEST PURPOSE	Validate that the allowed ODATA options do return the HTTP link header `rel="self"`.
TEST METHOD	1. Construct multiple SensorThings URLs each including a supported ODATA option that is not listed under the `odata_denied`. 2. Issue HTTP HEAD and GET requests on each URL. 3. Validate the contents of the returned HTTP header to include the link header `rel="self"` that includes the issued URL.

A.1.20. Example root page with denied ODATA option `$expand`

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": ["$expand"]
        }
    }
}
```

Listing A.13

A.1.21. Example HTTP response for a URL not including the denied ODATA option

```
GET /sta/v1.1/Things?$select=name HTTP/1.1
Host: localhost

HTTP/1.1 200 Ok
Content-type: application/json
Link: <http://hub.localhost>; rel="hub"
Link: <http://localhost/sta/v1.1/Things?%24select%3Dname>; rel="self"
```

Listing A.14

A.1.22. Tests for Topics Advertisement

ABSTRACT TEST A.13	
IDENTIFIER	`/conf/discovery/api-sta-websub-topics-discovery`
SUBJECT	/req/req-class-discovery/req-topics-discovery
REQUIREMENT	Requirement 11: `/req/req-class-discovery/req-topics-discovery`
LABEL	/conf/discovery/api-sta-websub-topics-discovery
TEST PURPOSE	Validate that the implemented STA-WebSub Conformance Class `Discovery` is listed on the root page.
TEST METHOD	1. Construct a URL to the root page. 2. Issue a HTTP HEAD and GET request on that URL. 3. Validate the contents of the returned document to include the implemented Conformance Class(es) in the `conformance` section of the root page: `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` 4. Validate the contents of the returned document to include the JSON object `https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` in the `serverSettings` including the key `topics_denied`. 5. Validate that the value of the key `topics_denied` is a JSON array that is either empty (“[]”) or contains a list of denied topics. 6. Validate that each value of the key `topics_denied` is compliant to `SERVICE_VERSION/RESOURCE_PATH_TO_AN_ENTITY`.

A.1.23. Example root page snippet advertising no denied ODATA options

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": []
        }
    }
}
```

Listing A.15

A.1.24. Example root page snippet advertising `v1.1/Observations` as a denied topic

```
{
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": ["v1.1/Observations"],
            "odata_denied": []
        }
    }
}
```

Listing A.16

A.1.25. Tests for Denied Topics Blacklisting

ABSTRACT TEST A.14	
IDENTIFIER	`/conf/discovery/api-sta-websub-topics-blacklisting`
SUBJECT	/req/req-class-discovery/req-topics-blacklisting
REQUIREMENT	Requirement 12: `/req/req-class-discovery/req-topics-blacklisting`
LABEL	/conf/discovery/api-sta-websub-topics-blacklisting
TEST PURPOSE	Validate that the denied topics as advertised on the root page are actually denied for discovery.
TEST METHOD	1. Construct a URL using a denied topic. 2. Issue a HTTP HEAD and GET request on that URL. 3. Validate the returned link headers to contain the `link rel="help"` and **not** `link rel="self"`. 4. Validate the contents of the returned HTTP header to include the link header `rel="hub"`.

A.1.26. Example root page snippet advertising `v1.1/Datastreams(4711)` as a denied topic

```
{
    "serverSettings": {
        "conformance": {
```

```
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": ["v1.1/Datastreams(4711)"],
            "odata_denied": []
        }
    }
}
```

Listing A.17

A.1.27. Example HTTP response for a URL including the denied topic `v1.1/Datastreams(4711)`

```
GET /sta/v1.1/Datastreams(4711) HTTP/1.1
Host: localhost

HTTP/1.1 200 Ok
Content-type: application/json
Link: <http://hub.localhost>; rel="hub"
Link: <http://localhost/sta/v1.1/help#topic_denied>; rel="help"
```

Listing A.18

A.1.28. Tests for no denied Topic Blacklisting

ABSTRACT TEST A.15	
IDENTIFIER	`/conf/discovery/api-sta-websub-topics-no-blacklisting`
SUBJECT	/req/req-class-discovery/req-topics-blacklisting
REQUIREMENT	Requirement 12: `/req/req-class-discovery/req-topics-blacklisting`
LABEL	/conf/discovery/api-sta-websub-topics-no-blacklisting
TEST PURPOSE	Validate that all topics are actually allowed for discovery.
TEST METHOD	1. Construct multiple URLs for accessing a topic. 2. Issue a HTTP HEAD and GET request on each URL. 3. Validate the returned link headers to contain the `link rel="self"` and **not** `link rel="help"`.

A.1.29. Example root page snippet advertising no denied topics

```
{
```

```
    "serverSettings": {
        "conformance": {
            "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/
discovery"
        },
        "https://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery":
{
            "topics_denied": [],
            "odata_denied": []
        }
    }
}
```

Listing A.19

A.1.30. Example HTTP request and response for a URL not including a denied topic

```
GET /sta/v1.1/Observations HTTP/1.1
Host: localhost

HTTP/1.1 200 Ok
Content-type: application/json
Link: <http://hub.localhost>; rel="hub"
Link: <http://localhost/sta/v1.1/Observations>; rel="self"
```

Listing A.20

B

ANNEX B (INFORMATIVE) STA-WEBSUB HUB IMPLEMENTATION GUIDANCE

ANNEX B (INFORMATIVE) STA-WEBSUB HUB IMPLEMENTATION GUIDANCE

This appendix provides guidance as to how to implement a STA-WebSub Hub interoperating with a SensorThings API v1.1 service.

NOTE: This appendix uses the natural language word “MUST” to indicate de-facto implementation requirements for a STA-WebSub Hub. The formal identification of requirements using the word “SHALL” is not used because the STA-WebSub Hub is not the standardization target!

B.1. Callback Authentication

An implementation MUST accept the `hub.api_key` or `hub.x_api_key` with a subscription request. If both parameters (the `hub.api_key` **and** the `hub.x_api_key`) are used, the implementation MUST deny the subscription and return a HTTP response status code 400.

- If the subscription request includes the `hub.api_key` parameter, the Hub MUST add the HTTP header `Api-Key` to the request send to the Subscriber’s callback
- If the subscription request includes the `hub.x_api_key` parameter, the Hub MUST add the HTTP header `X-Api-Key` to the request send to the Subscriber’s callback

The `hub.api_key` and `hub.x_api_key` parameters MUST be used exclusively and SHOULD only be specified with static callback URLs. The parameter value MUST be less than 200 bytes in length.

B.2. X-Api-Key and X-Hub-Signature

One fundamental runtime functionality for a Hub is its ability to deliver notification content to all Subscribers. The W3C WebSub offers the use of HMAC (Hash-based Message Authentication Code), allowing Subscribers to validate the authenticity of the received content and ensure it comes from the expected Hub. By leveraging HMAC, Subscribers can also verify

content integrity, which is crucial when using non-secure communication, such as HTTP instead of HTTPS. The W3C WebSub supports the transmission of the HMAC through the HTTP header `X-Hub-Signature`.

However, the use of HMAC signatures imposes a burden on the CPU and memory of both the Hub and the Subscribers when calculating the HMAC value. This burden also necessitates caching (temporary storage) of large content at both the Hub and Subscriber implementations. As a result, the use of HMAC prevents Subscribers from streaming the received content directly into connected workflow processing. On the Hub side, caching does not introduce a significant resource burden, as an implementation would likely cache the content before distributing it to all Subscribers.

Validating the HMAC requires that a Subscriber receives the entire content, which creates a potential denial-of-service vulnerability. If an attacker sends large fraudulent content at high frequency using numerous parallel requests, the Subscriber must process the entire content to determine its authenticity, which may ultimately lead to denial of processing.

The W3C WebSub recommends using a random callback URL that is refreshed by the Subscriber whenever a subscription is updated. This practice offers equivalent protection for the callback endpoint compared to using an API key. When a callback endpoint is associated with a single Hub and operates over HTTPS, the use of HMAC becomes unnecessary.

In cases where refreshing callback URLs is not feasible or too complex, STA-WebSub Hub supports API key authentication as an alternative. The use of an API key compensates for the lack of random callback URLs. Furthermore, an API key, combined with HTTPS callback URLs, provides the same level of authenticity and integrity as HMAC, but without the need to calculate HMAC on the Hub side and validate it on the Subscriber side. This approach reduces the resource burden on both the Hub and Subscriber, which is particularly important for Hub deployments on the edge.

Similar to how `hub.secret` shares the secret for HMAC generation between a Subscriber and a Hub, a STA-WebSub subscription may include the parameters `hub.api_key` or `hub.x_api_key` to activate API key usage at the Hub. These parameters prompt the Hub to add the HTTP headers `Api-Key` or `X-Api-Key` when distributing content to the Subscriber.

B.3. Subscription

The STA-WebSub Hub, which is linked to a SensorThings API service, supports the W3C subscribe/unsubscribe protocol and converts a subscription topic URL (`hub.topic`) into an MQTT topic. A compliant implementation must be capable of transforming the absolute HTTP(S) request URL into the corresponding MQTT topic for the associated SensorThings API service.

The Hub utilizes the MQTT protocol to handle subscriptions and unsubscriptions with the MQTT broker that corresponds to the SensorThings API service.

An implementation of the STA-WebSub Hub must convert the HTTP discovery URL into an MQTT topic by omitting the SensorThings API `baseUrl`. For instance, in a deployment where

`baseUrl=http://localhost:8080/mysta`, the discovery URL `http://localhost:8080/mysta/v1.1/Observations` would result in the MQTT topic `v1.1/Observations`.

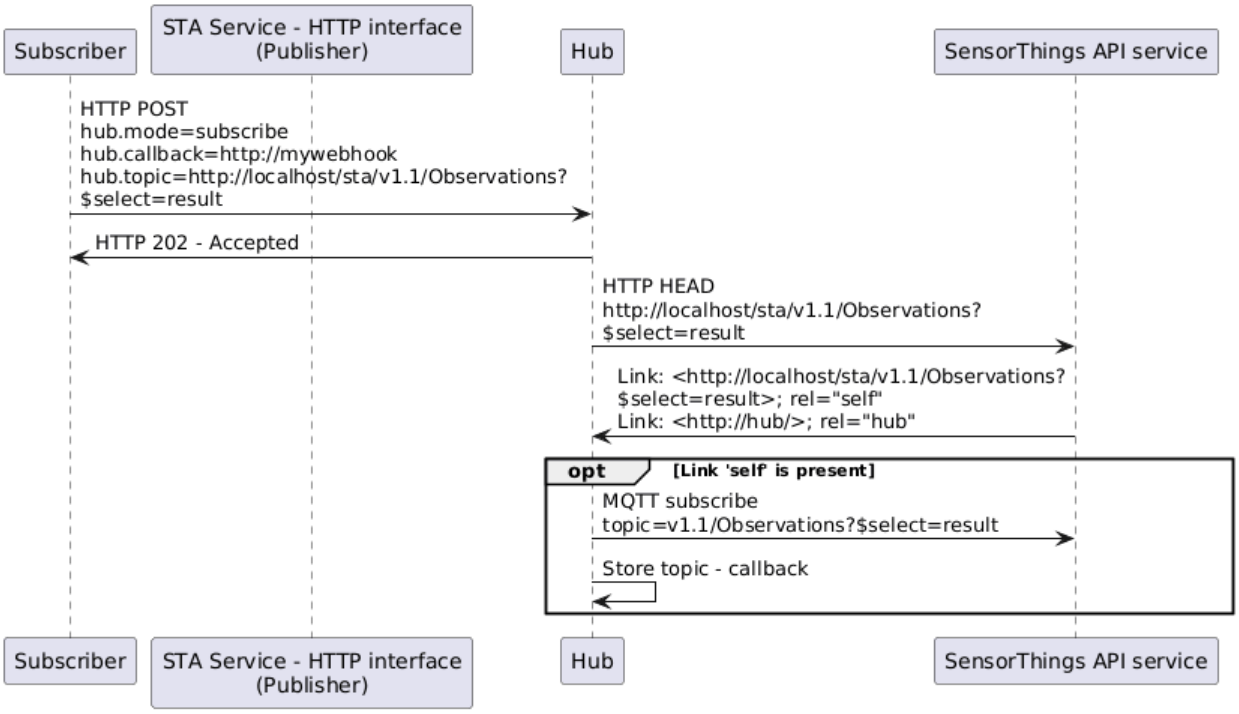


Figure B.1 — Subscription Handling in the STA-WebSub Hub

As defined in the W3C WebSub, a Subscriber sends a subscription request to the Hub using the URL from the `<Link/> rel="hub"`. The subscription topic (`hub.topic`) is identical to the URL returned with the `<Link/> rel="self"`.

Upon receiving the subscription request, it is highly recommended that the Hub validates the Subscriber's intent as defined in the W3C WebSub specification. For a STA-WebSub Hub, it is also advisable to check the subscription's validity with the publisher using the discovery protocol. If successful (indicated by the inclusion of `Link rel="self"` in the discovery response), the Hub should then subscribe to the SensorThings API service via MQTT, transforming the request URL into an MQTT topic.

It is recommended to use the HTTP HEAD method for discovery requests, as this approach helps avoid unnecessary data transfer.

For new subscriptions or when renewing an existing one (when the Hub receives a subscribe request for an already active subscription), the implementation must set or update the API key and use this value for subsequent content distribution to the Subscriber.

B.4. Notification

The STA-WebSub Hub must support receiving MQTT notifications from the MQTT broker associated with the SensorThings API service.

The implementation must listen for notifications from the MQTT broker of the SensorThings API service.

Once a notification is received, the implementation must distribute the content of the notification to all subscribers using HTTP(S), as defined by the W3C WebSub Recommendation.

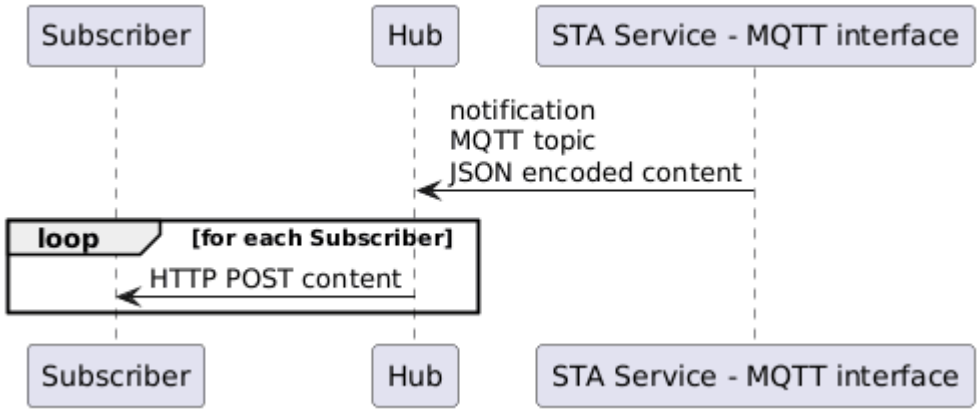


Figure B.2 — MQTT notification delivery by the STA-WebSub Hub

If the subscription was initiated using the `hub.api_key` or `hub.x_api_key`, the implementation must include the `Api-Key` or `X-Api-Key` in the HTTP request sent to the Subscriber's callback URL (Webhook).

Similarly, the implementation must add the `X-Hub-Signature` header if the subscription was created using the `hub.secret` parameter.

B.5. Implementations

Prototype implementations that illustrate a STA-WebSub system are available as open source.

- The STA-WebSub Hub is available as open source from Github: https://github.com/securedimensions/STA-WebSub-Hub.
- The W3C discovery protocol extension for the SensorThings API service is implemented as a plugin to FROST-Server:
  - STA-WebSub plugin: https://github.com/securedimensions/FROST-Server-WebSub; and

    - The FROST-Server: https://github.com/FraunhoferIOSB/FROST-Server.

- A simple and generic WebSub Subscriber is available as open source from Github: https://github.com/securedimensions/STA-WebSub-Subscriber.
- A simple and generic client for testing the STA-WebSub discovery protocol is available as open source: https://github.com/securedimensions/STA-WebSub-Discovery.

NOTE:The key work for crafting this OGC Standard was undertaken in 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.

These implementations are provided for education and **must not** be used in production! Improvements via pull-requests are welcome.

C

ANNEX C (INFORMATIVE) REVISION HISTORY

ANNEX C (INFORMATIVE) REVISION HISTORY

Table C.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2024-10-08	0.1	Andreas Matheus	all	initial version
2024-12-11	0.2	Andreas Matheus	all	improvements based on implementations
2024-12-19	0.3	Andreas Matheus	all	restructuring to meet OGC standards structure
2025-01-14	0.4	Andreas Matheus	all	structure improvement; Appendix A first draft
2025-01-17	0.5	Andreas Matheus	all	First draft of normative section and completion of Appendix A
2025-01-23	0.6	Andreas Matheus	all	Incorporating feedback from Joan Maso
2025-01-31	0.7	Andreas Matheus	all	Making the normative chapters (STA-WebSub Requirements and Annex A) compliant with OGC Mod-Spec
2025-02-04	0.8	Andreas Matheus	Chapter 7 and Annex A	Incorporating feedback from Joan Maso and Hylke van der Schaaf
2025-02-25	0.9	Andreas Matheus	all	Incorporating editorial improvements from Carl Reed
2025-08-28	0.10	Andreas Matheus	all	Final edits
2026-04-01	1.0	Andreas Matheus	all	Final edits for publication as OGC Standard

BIBLIOGRAPHY

BIBLIOGRAPHY