OGC® DOCUMENT: 24-032

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Open Geospatial Consortium

OGC SENSORTHINGS API EXTENSION: WEBSUB ASYNCHRONOUS MESSAGING STANDARD

STANDARD
Implementation

DRAFT

Version: 0.9
Submission Date: 2024-07-03
Approval Date: 2025-02-25
Publication Date: 2029-12-31
Editor: Andreas Matheus,

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I ABSTRACT

The STA-WebSub — SensorThings API extension WebSub — Standard defines an additional SensorThings API capability that supports the distribution of updates via HTTP(S) as defined in the W3C WebSub Recommendation (see W3C WebSub Recommendation).

A subscriber can fetch base data from a SensorThings API service and then use the same identical URL to subscribe for updates. This avoids the need for any subscriber to poll a SensorThings API service. This is because any updates — according to the URL used — get submitted to the subscriber's Webhook when the update event is triggered. The use of this WebSub extension is also easy to integrate into existing systems. This is because producers only need to set up a W3C WebSub compliant Hub that listens for subscription updates via MQTT. Consumers only need to setup a WebSub Subscriber to receive updates via WebHook. The SensorThings API MQTT protocol does not need to be exposed to the subscriber. The MQTT protocol can remain internal (hidden) between the SensorThings API service and the associated STA-WebSub Hub(s). Any SensorThings API service that extends the MQTT topic pattern to include filter and expand capabilities via ODATA query parameters will allow to subscribe for updates by defining actual triggering conditions using `$filter`. Using the ODATA query parameter `$select` and `$expand` also supports the ability of the subscriber to get exactly the data and structure as it is fit for purpose.

The use of STA-WebSub improves the flexible use of SensorThings API services for building asynchronous workflows. The ability to define trigger conditions and to specify the event data structure that is pushed over HTTP(S) POST enables a fit-for-purpose processing of events using workflows. Access control can be implemented in the discovery functionality which controls the subscription based on business or access control policies.

II KEYWORDS

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

ogcdoc, OGC document, OGC Standard, API, SensorThings, WebSub

III PREFACE

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

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

IV SECURITY CONSIDERATIONS

Any event processing system is susceptible to denial of service attacks. This can happen when too many clients subscribe to topics with high frequency updates (`chatty topics`) or too many updates are received. For example, subscription to the MQTT topic `v1.1/Observations` brings the danger that the SensorThings API service cannot deliver all events to all subscribed clients when the update frequency gets too high. Additionally, the subscriber may get flooded with event data sent by the Hub.

Any implementation that supports the use of ODATA query parameters is particularly susceptible to denial of service due to the complexity of an ODATA query. An attacker could achieve a denial of service attack by subscribing using a very complex topic URL including an ODATA query. To mitigate such an attack, an implementation should carefully analyze the query parameters of a topic URL before accepting the subscription request.

Any SensorThings API service should deny MQTT update subscriptions that do not origin from an associated Hub. This guarantees that an attacker cannot simply bypass the Hub and swamp the SensorThings API service directly with bogus update subscriptions.

The proper network separation of SensorThings API service and the Hub, such as in a fast private network, enables establishing a clever load distribution between a SensorThings API service and as many Hubs as needed. This self-adaptivity to scaling mitigates the likelihood for denial of service attacks even with many subscribed clients and complex topics.

In any case, the Hub must validate a subscription with the STA-WebSub Service. This can be achieved via the regular discovery request (HTTP Head request with the topic URL).

To prevent bogus subscriptions, a Hub may require authentication for the subscription management.

The W3C WebSub recommends that the subscriber's callback URL is 'not easily guessable' and that the callback URL is refreshed whenever a subscription is renewed. Following this recommendation makes the use of other authentication mechanisms, such as the use of an API key, unnecessary. An API key is a unique identifier that authenticates a user or application to an API instance. However, for computing environments where it is not possible or too difficult to follow this recommendation, the use of `API-Key` or `X-API-Key` HTTP header can protect a static callback endpoint URL. Likewise, when refreshing the callback URL, the subscriber updating the API key value with a strong random value each time a subscription is renewed it is strongly recommended.

Leveraging an API key over HTTPS callback URLs may be preferred over the use of `X-Hub-Signature` when the notification data size is large. The clear advantage is that neither the Hub nor the Subscriber must calculate a hash over the notification data. Also, the use of API key over HTTPS is equivalent to HMAC. Different API keys for subscriptions identifies the Hub, thereby providing authenticity. The use of 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	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

This OGC SensorThings API Extension: WebSub Asynchronous Messaging Standard defines how to extend the OGC SensorThings API v1.1 implementation towards asynchronous messaging over HTTP by adopting 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. Topic URL

a URL used by a Subscriber when subscribing to a Hub. The SensorThings API service returns the topic URL in the HTTP Link header `<Link "topic URL">; rel="self"` for a request URL that is allowed for subscription.

4.6. MQTT Topic

a string used to subscribe to the MQTT broker for receiving update events. The MQTT topic is determined 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

`http://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.

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

A STA-WebSub System 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 — of which the publisher can be split into two components: The HTTP interface of the SensorThings API service and the MQTT broker.

This STA-WebSub extension to SensorThings API only defines the requirements for the HTTP interface to implement the W3C WebSub Discovery protocol. As such, there are no impacts on backwards compatibility.

The requirements of the Subscriber and Hub are also discussed in this document, but are not standardized. However, Appendix B provides guidance as to how to implement a STA-WebSub Hub for 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 addresses how to implement asynchronous processing over the Web. First, the discovery protocol allows a Subscriber to check with a Publisher if self-defined topics can be used for subscription. Second, the subscription protocol defines how a Subscriber can interact with a hub to register a topic for receiving notifications afterwards.

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

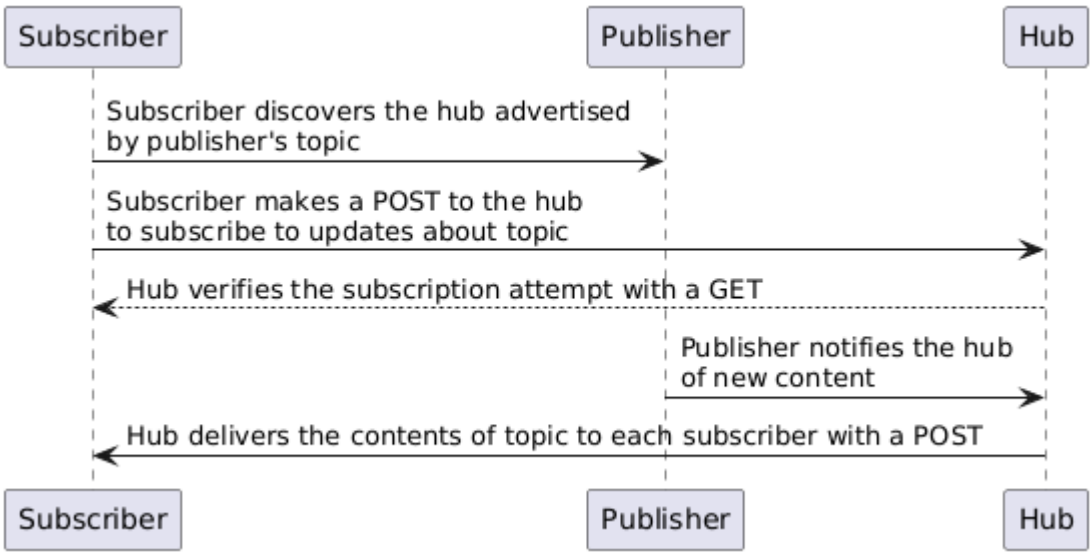


Figure 1 — W3C WebSub Flow Diagram

The notification protocol is deliberately left 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 and the 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.

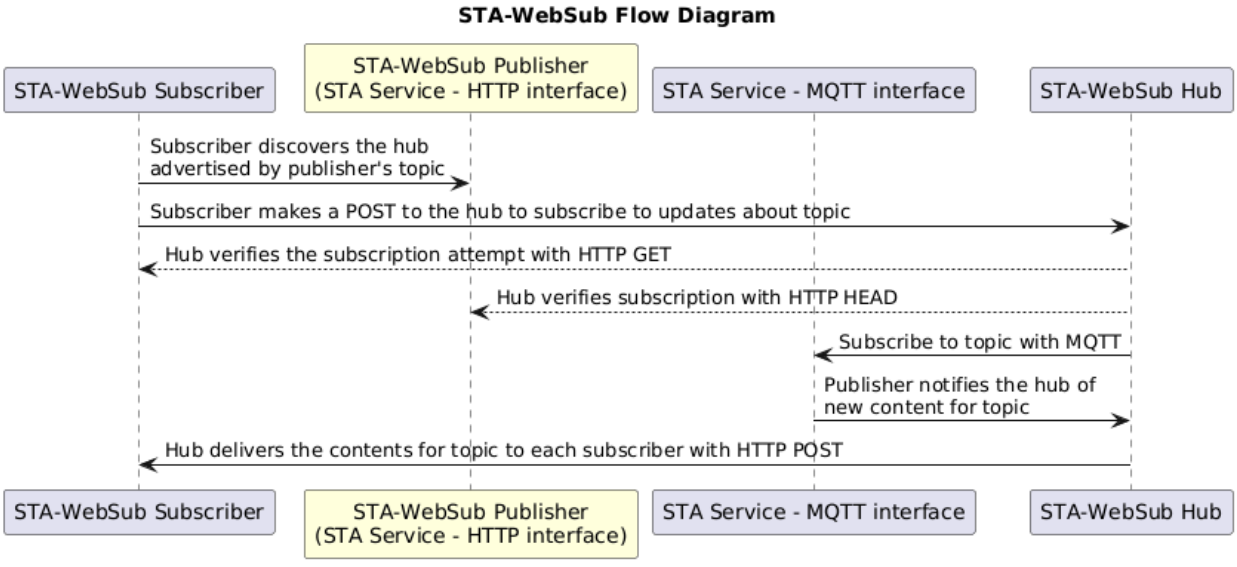
A STA-WebSub system consists of four components:

- SensorThings HTTP interface:
  - The W3C discovery protocol must be implemented.
  - The STA-WebSub error handling must be implemented.
- SensorThings MQTT broker:
  - The Hub subscribes (and unsubscribes) for updates on MQTT topics
  - 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.
  - 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 GET or HEAD requests to the SensorThings API service — HTTP interface
  - A URL that can be used for subscription with the Hub (via `hub.topic`) can be obtained from the HTTP Link header `rel="self"` of the discovery response
  - A subscriber may use an `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 as defined in the W3C WebSub Recommendation. The notification event and content is based on MQTT as defined in the SensorThings API Standard.

To make STA-WebSub work, the subscriber, publisher and Hub needs 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 implementations of the publisher and the Hub. But only the part of the publisher (yellow box) is considered normative here, as the SensorThings API service is the

standardization target. To ensure a working solution, the Appendix B provides guidance for the implementation specific aspects to adapt a W3C Hub for STA-WebSub.

Figure 2 — STA-WebSub Flow Diagram

The use of W3C WebSub does not predefine topics. Instead, the subscriber can discover if a topic (URL) can be used for subscription. This flexibility is important for STA-WebSub to support the subscriber for defining

- the actual trigger for a update event, and
- the data structure for the data delivered with an update event.

Any implementation of the WebSub extension must support a MQTT topic pattern as defined in the SensorThings API v1.1 Standard extended by an ODATA query. A SensorThings API service implementation that supports ODATA options with MQTT topics offers the subscriber to use `$filter` for specifying the event trigger and `$select` and `$expand` to specify the content data structure. The ability to craft notification conditions in combination to compose exactly the data structure needed for a notification is a powerful ability to trigger fit for purpose workflow executions connected 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

To extend the W3C WebSub protocol to support STA-WebSub, certain functional requirements needed to be defined regarding discovery, subscription and notification.

6.3.1. Discovery

A STA-WebSub compliant SensorThings API HTTP interface (aka publisher) would support the W3C WebSub discovery by adding the `Link` headers `rel="hub"` and `rel="self"` to the HTTP response. W3C WebSub further requires that the `Link` headers are returned either as HTTP response headers or inline to a XHTML encoded response. As the typical response encoding for an implementation of the SensorThings API is usually either JSON or GeoJSON and not XHTML, the STA-WebSub extension requires that the `Link` headers are returned as HTTP response headers. Further,the W3C WebSub anticipates that these discovery links will be returned on a HTTP GET or HEAD request. To meet the HTTP method requirement for discovery, the STA-WebSub extension requires that a publisher implementation supports the HTTP HEAD method in addition to the already supported HTTP GET method.

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

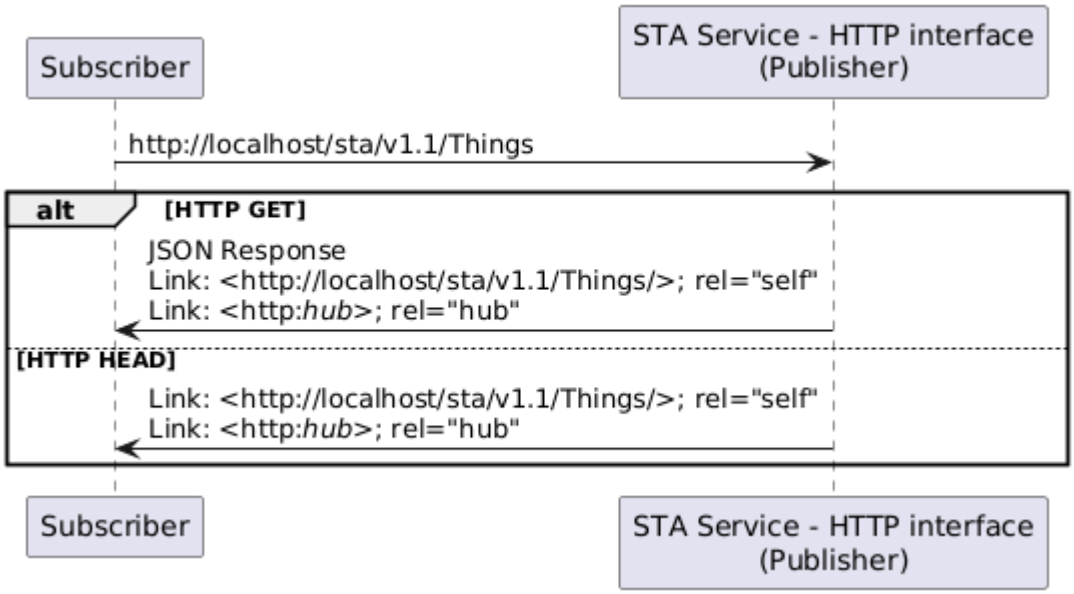


Figure 3 — Discovery with URL for supported subscription

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` that transforms into a MQTT topic which may not be accepted by the SensorThings API MQTT broker. Which topic URLs are accepted is deployment specific. For example, the subscription to `/Observations` may produce a too high load. Another example is the use of not supported / not allowed ODATA options like `$expand` or `$filter` which may cause that no `Link rel="self"` is returned. The missing `Link rel="self"` header implies that subscription for such a topic URL is not possible. This behavior is perfectly compliant with the WebSub Recommendation. However, any subscriber (user or service/process) may wonder why the discovery response does not include the `Link rel="self"` header.

Technically, the fact that no `Link rel="self"` is returned is not an error. Therefore, the use of HTTP 4xx status codes is not appropriate. Also, the W3C WebSub does not specify any error handling. But to support a subscriber, there should be guidance as to why the link header `rel="self"` is missing.

This STA-WebSub Standard introduces the use of the `Link rel="help"` header. The URL for this relationship must point to a (static) help page that explains why a subscription to the topic URL is not possible.

The following sequence diagram illustrates the discovery of a URL that is not supported for subscription: `http://localhost/sta/v1.1/Observations`

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

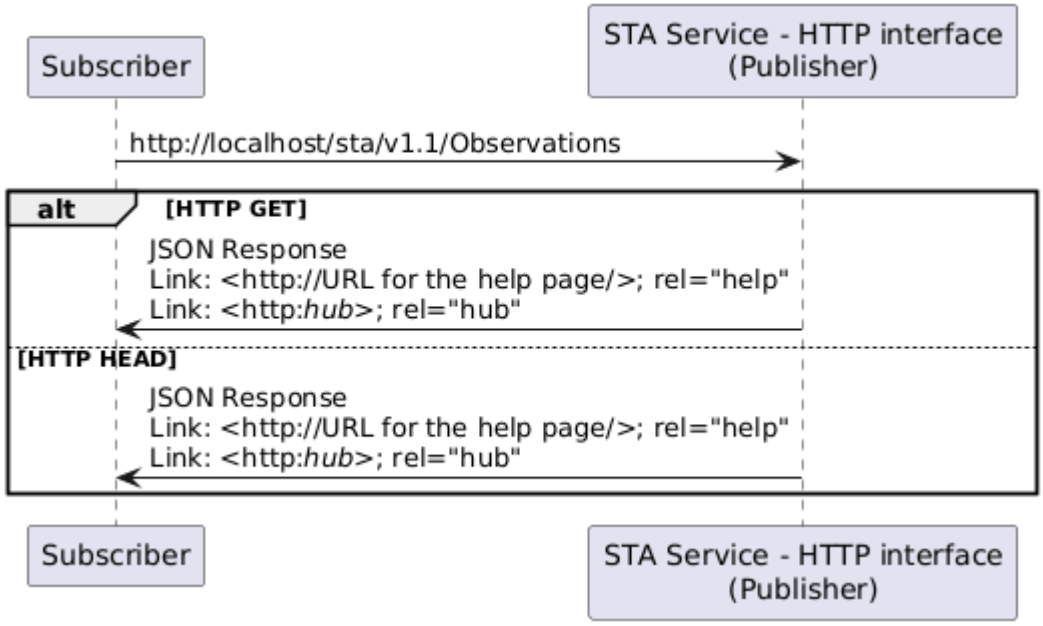


Figure 4 — Discovery with URL not supported for subscription due to topic restriction

For this URL, the implementation does not return the `<Link/>; rel="self"` but the `<Link/>; rel="help"` header.

The following sequence diagram illustrates the discovery for a URL that is not supported for subscription due to disallowed ODATA options: `http://localhost/sta/v1.1/Datastreams(4711)/Observations?$expand=FeatureOfInterest`

NOTE 2: It is assumed that the ODATA option `$expand` is blacklisted.

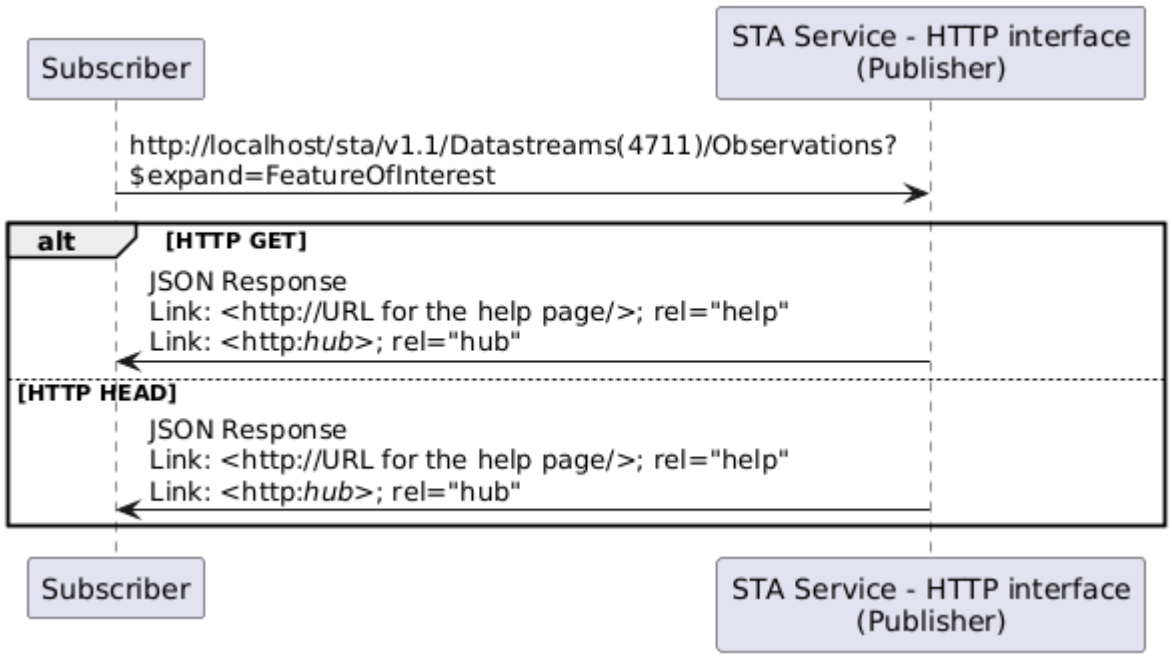


Figure 5 — Discovery with URL not supported for subscription due to ODATA restriction

For this URL, the implementation does not return the `<Link/>; rel="self"` but the `<Link/>; rel="help"` header.

6.3.3. Subscriptions

A STA-WebSub Hub implementation is capable of transforming the W3C WebSub `hub.topic` expressed as a HTTP(S) URL into a MQTT topic pattern accepted by the MQTT broker associated with the SensorThings API service. As the SensorThings API service and the Hub are working in close relation, this transformation should not be too difficult.

Preventing that a subscription to an unsupported MQTT topic is possible, the Hub must use the discovery protocol and deny the request if the discovery response does not include the `Link rel="self"` header.

In case the Hub receives an unsubscribe request from the subscriber, the Hub must verify the intent with the subscriber and unsubscribe from the MQTT topic with the associated SensorThings API service.

6.3.4. Notifications

Once the Hub has subscribed to a MQTT topic, it awaits MQTT notifications from the MQTT broker of the SensorThings API service. In case of a notification event (topic + content), the Hub delivers the content to the subscribed callback URLs (the subscribers' Webhooks) using HTTP POST.

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`:

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

```
        }
    }
}
```

6.5. Benefits of STA-WebSub

The OGC 18-088 allows MQTT clients to subscribe to update events. This means that update events are delivered from the MQTT broker to all subscribed clients. The STA-WebSub extension, as defined in this OGC Standard, supports the distribution of the MQTT events via HTTP(S). Therefore, the MQTT broker does not send the update events 1→n to subscribers, it is the Hub that does that using HTTP(S). This separation of duty brings important improvements regarding use, security and scalability:

- The use of the STA-WebSub extension makes the MQTT protocol internal between the MQTT broker of the SensorThings API service and the associated Hub(s). This allows controlling MQTT subscriptions origin at the associated STA-WebSub Hub. Also, the use of discovery policies allow to implement flexible and fine grained access control regarding the subscriptions done by the Hub.
- The fact that the MQTT protocol is internal between the Hub and the SensorThings Service simplifies the use for subscribers to well-known infrastructure patterns like Webhook, essentially using a W3C WebSub compliant HTTP(S) endpoint listening for GET and POST requests.
- The separation of duty for sending update 'content' to subscribers between the SensorThings API and the Hub improves scalability. The SensorThings API service only delivers the topic updates to associated Hub(s) using MQTT. The Hub(s) then optionally processes the MQTT message and distributes the content to subscribers using well understood cloud-scaling code stacks.
- The ability that subscribers can determine the notification conditions (i.e. using `$filter`) and the data structure of the notification (i.e. using `$select` and `$expand`) improves the usability over predefined MQTTP topics. How flexible a subscriber can get is controlled by the discovery functionality.
- The ability to do subscriptions/publications using HTTP implemented by the Webhook enables for implementing asynchronous messaging by a pure HTML5/JavaScript web client that receives the updates via web sockets without setting up any extensions in the web browsers.

7

SENSORTHINGS API – WEBSUB EXTENSION (NORMATIVE)

7 SENSORTHINGS API — WEBSUB EXTENSION (NORMATIVE)

As outlined in chapter Clause 6, a system leveraging the SensorThings APIWebSub Extension (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.

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

The STA-WebSub extension defines the following requirements class:

- Discovery Requirements Class (mandatory)

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	`http://www.opengis.net/spec/sta-websub/1.0/req/discovery`
TARGET TYPE	SensorThings API v1.1 Service
NORMATIVE STATEMENTS	Requirement 1-1: `/req/req-class-discovery/req-req-http-methods` Requirement 2: `/req/req-class-discovery/req-link-hub` Requirement 3: `/req/req-class-discovery/req-link-self` Requirement 4: `/req/req-class-discovery/req-link-help` Requirement 5: `/req/req-class-discovery/req-odata-support` Requirement 6: `/req/req-class-discovery/req-odata-discovery` Requirement 7: `/req/req-class-discovery/req-odata-blacklisting` Requirement 1-8: `/req/req-class-discovery/req-odata-no-blacklisting` Requirement 8: `/req/req-class-discovery/req-topics-discovery` Requirement 9: `/req/req-class-discovery/req-topics-blacklisting` Requirement 1-11: `/req/req-class-discovery/req-topics-no-blacklisting`

7.1.1. Requirement HTTP Methods

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

REQUIREMENT 1	
IDENTIFIER	`/req/req-class-discovery/req-http-methods`
A	An implementation SHALL support discovery via HTTP GET method.
B	An implementation SHALL support discovery via HTTP HEAD method.

7.1.2. Requirement Link Header 'hub'

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

REQUIREMENT 2	
IDENTIFIER	`/req/req-class-discovery/req-link-hub`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/sta-websub/1.0/req/discovery`
A	An implementation SHALL return the `<Link/>; rel="hub"` HTTP header to indicate support for WebSub.

7.1.3. Requirement Link Header 'self'

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

REQUIREMENT 3	
IDENTIFIER	`/req/req-class-discovery/req-link-self`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/sta-websub/1.0/req/discovery`
A	An implementation SHALL use the `<Link/>; rel="self"` and`<Link/>; rel=”help”` HTTP header mutually exclusive.

REQUIREMENT 3	
B	An implementation SHALL return the `<Link/>; rel="self"` HTTP header to indicate the ability for subscription.
C	An implementation SHALL not return the `<Link>; rel="self"` HTTP header if the request URL either contains a denied topic or at least one denied ODATA option. In this this case, the implementation SHALL return the `<Link>; rel="help"`.
D	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.
E	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.

7.1.4. 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 4	
IDENTIFIER	`/req/req-class-discovery/req-link-help`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/sta-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.

7.1.5. Requirement ODATA query options support

REQUIREMENT 5	
IDENTIFIER	`/req/req-class-discovery/req-odata-support`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/sta-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:

REQUIREMENT 5	
	• 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

7.1.6. Requirement ODATA support advertisement

REQUIREMENT 6	
IDENTIFIER	/req/req-class-discovery/req-odata-discovery
INCLUDED IN	Requirements class 1: http://www.opengis.net/spec/sta-websub/1.0/req/discovery
A	An implementation SHALL advertise the support for ODATA query parameter by publishing the string http://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery in the serverSettings section of the root page.

7.1.7. Requirement ODATA Blacklisting

A STA 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 7	
IDENTIFIER	/req/req-class-discovery/req-odata-blacklisting
INCLUDED IN	Requirements class 1: http://www.opengis.net/spec/sta-websub/1.0/req/discovery
A	An implementation SHALL advertise the existence of denied ODATA options by adding the following key to the root page: http://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: http://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": {
        "http://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": {
        "http://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": {
        "http://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery": {
            "odata_denied": []
        }
    }
}
```

7.1.8. Requirement Topics Blacklisting

The support for STA-WebSub can be determined from the root page as the relevant conformance classes are listed.

REQUIREMENT 8	
IDENTIFIER	`/req/req-class-discovery/req-topics-discovery`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/sta-websub/1.0/req/discovery`
A	An implementation SHALL advertise support for topics blacklisting by publishing the string `http://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": {
        "http://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery": {
            "topics_denied": <JSON Array of denied topics>
        }
```

```
    }
}
```

REQUIREMENT 9	
IDENTIFIER	`/req/req-class-discovery/req-topics-blacklisting`
INCLUDED IN	Requirements class 1: `http://www.opengis.net/spec/sta-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: `http://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` that includes the key `topics_denied`. This key SHALL contain a JSON array for all denied topics.
C	In case no topics are denied,an implementation SHALL advertise the non-existence of denied ODATA options by adding the following key to the root page: `http://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": {
        "http://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": {
        "http://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 [_websub].

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)

A

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: `http://www.opengis.net/spec/sta-websub/1.0/req/discovery`
TARGET TYPE	SensorThings API v1.1 Service
LABEL	http://www.opengis.net/spec/sta-websub/1.0/conf/discovery
CONFORMANCE TESTS	Abstract test A.1: `/conf/discovery/api-sta-websub-http-get` Abstract test A.2: `/conf/discovery/api-sta-websub-http-head` Abstract test A.4: `/conf/discovery/api-sta-websub-link-header-help` Abstract test A.3: `/conf/discovery/api-sta-websub-link-header-self` Abstract test A.5: `/conf/discovery/api-sta-websub-odata-support` Abstract test A.6: `/conf/discovery/api-sta-websub-odata-discovery` Abstract test A.7: `/conf/discovery/api-sta-websub-odata-blacklisting` Abstract test A.8: `/conf/discovery/api-sta-websub-odata-no-blacklisting` Abstract test A.9: `/conf/discovery/api-sta-websub-topics-discovery` Abstract test A.10: `/conf/discovery/api-sta-websub-topics-blacklisting` Abstract test A.11: `/conf/discovery/api-sta-websub-topics-no-blacklisting`

A.1.1. HTTP Methods

ABSTRACT TEST A.1

IDENTIFIER	`/conf/discovery/api-sta-websub-http-get`
SUBJECT	/req/req-class/discovery
REQUIREMENTS	Requirement 1: `/req/req-class-discovery/req-http-methods` Requirement 2: `/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 and that the response headers include the Link headers `hub`. **NOTE 1:** The Link header `hub` must always be returned even for a topic URL that is not allowed for subscription. But for this test, it is important to test that the HTTP method is supported and for that, the existence of the `hub` header is sufficient.

ABSTRACT TEST A.2

IDENTIFIER	`/conf/discovery/api-sta-websub-http-head`
SUBJECT	/req/req-class/discovery
REQUIREMENTS	Requirement 1: `/req/req-class-discovery/req-http-methods` Requirement 2: `/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 and that the response headers include the Link headers `hub`. **NOTE 2:** The Link header `hub` must always be returned even for a topic URL that is not allowed for subscription. But for this test, it is important to test that the HTTP method is supported and for that, the existence of the `hub` header is sufficient.

A.1.2. Link Headers

ABSTRACT TEST A.3	
IDENTIFIER	`/conf/discovery/api-sta-websub-link-header-self`
SUBJECT	/req/req-class/discovery/req-discovery-self
REQUIREMENT	Requirement 3: `/req/req-class-discovery/req-link-self`
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 a topic URL that is **allowed** for subscription. A topic URL does not include a query string. 2. Issue a HTTP GET or HEAD request on that topic URL. 3. Validate that the response includes the Link header `hub` and `self`. Also validate that the Link header `help` is not present! **NOTE 1:** To determine which topics **are allowed**, examine the root page and check the `topics_denied` array.

ABSTRACT TEST A.4	
IDENTIFIER	`/conf/discovery/api-sta-websub-link-header-help`
SUBJECT	/req/req-class/discovery/req-discovery-help
REQUIREMENT	Requirement 4: `/req/req-class-discovery/req-link-help`
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. 2. Issue a HTTP GET or HEAD request on that topic URL. 3. Validate that the response includes the Link header `hub` and `help`. In particular validate that the Link header `self` is not present!

ABSTRACT TEST A.4

NOTE 2: To determine which topic URLs **are not allowed**, examine the root page. The cause for 'not allowed' SHALL be based on `topics_denied` or `odata_denied`.

A.1.3. Example for topics and ODATA restrictions

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

Listing A.1

A.1.4. Example HTTP request and response for a URL including the denied topic `v1.1/Observations`

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

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

Listing A.2

A.1.5. 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: example.com

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

Listing A.3

A.1.6. Tests for ODATA Support

ABSTRACT TEST A.5	
IDENTIFIER	`/conf/discovery/api-sta-websub-odata-support`
REQUIREMENT	Requirement 5: `/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). 2. Issue a HTTP GET request on each URL. 3. Validate the returned content to not contain errors.

A.1.7. Tests for ODATA Support Advertisement

ABSTRACT TEST A.6	
IDENTIFIER	`/conf/discovery/api-sta-websub-odata-discovery`
SUBJECT	/req/req-class/discovery
REQUIREMENT	Requirement 6: `/req/req-class-discovery/req-odata-discovery`
LABEL	/conf/discovery/api-sta-websub-odata
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 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: `http://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` 4. Validate the contents of the returned document to include the JSON object `http://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` in the `serverSettings` including the key `odata_denied`. 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.1.8. Example root page snippet advertising no denied ODATA options

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

Listing A.4

A.1.9. Example root page snippet advertising $expand as a denied ODATA option

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

Listing A.5

A.1.10. Tests for ODATA Options Blacklisting

ABSTRACT TEST A.7	
IDENTIFIER	`/conf/discovery/api-sta-websub-odata-blacklisting`
SUBJECT	/req/req-class/discovery
REQUIREMENT	Requirement 7: `/req/req-class-discovery/req-odata-blacklisting`
LABEL	/conf/discovery/api-sta-websub-odata
TEST PURPOSE	Validate that the disallowed ODATA options do not return the HTTP link header `rel="self"`.

ABSTRACT TEST A.7	
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"`.

A.1.11. Example root page with denied ODATA option `$expand`

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

Listing A.6

A.1.12. Example HTTP response for a URL including the denied ODATA option `$expand`

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

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

Listing A.7

A.1.13. Tests for no Blacklisting of ODATA Options

ABSTRACT TEST A.8	
IDENTIFIER	`/conf/discovery/api-sta-websub-odata-no-blacklisting`
SUBJECT	/req/req-class/discovery
REQUIREMENT	Requirement 7: `/req/req-class-discovery/req-odata-blacklisting`

ABSTRACT TEST A.8	
LABEL	/conf/discovery/api-sta-websub-odata
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.1.14. Example root page with denied ODATA option `$expand`

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

Listing A.8

A.1.15. Example HTTP response for a URL not including the denied ODATA option

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

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

Listing A.9

A.1.16. Tests for Topics Advertisement

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

ABSTRACT TEST A.9	
SUBJECT	/req/req-class-discovery/req-discovery
REQUIREMENT	Requirement 8: `/req/req-class-discovery/req-topics-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 the contents of the returned document to include the implemented Conformance Class(es) in the `conformance` section of the root page: `http://www.opengis.net/spec/sensorthings-websub/1.0/conf/discovery` 4. Validate the contents of the returned document to include the JSON object `http://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.

A.1.17. Example root page snippet advertising no denied ODATA options

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

Listing A.10

A.1.18. Example root page snippet advertising `v1.1/Observations` as a denied topic

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

```
        }
    }
}
```

Listing A.11

A.1.19. Tests for Denied Topics Blacklisting

ABSTRACT TEST A.10	
IDENTIFIER	`/conf/discovery/api-sta-websub-topics-blacklisting`
SUBJECT	/req/req-class-discovery
REQUIREMENT	Requirement 9: `/req/req-class-discovery/req-topics-blacklisting`
LABEL	/conf/discovery/api-sta-websub-landing-page
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 GET request on that URL. 3. Validate the returned link headers to contain the `link rel=help"` and **not** `link rel="self"`.

A.1.20. Example root page snippet advertising `v1.1/Datastreams(4711)` as a denied topic

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

Listing A.12

A.1.21. Example HTTP response for a URL including the denied topic `v1.1/Datastreams(4711)`

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

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

Listing A.13

A.1.22. Tests for no denied Topic Blacklisting

ABSTRACT TEST A.11	
IDENTIFIER	`/conf/discovery/api-sta-websub-topics-no-blacklisting`
SUBJECT	/req/req-class-discovery
REQUIREMENT	Requirement 9: `/req/req-class-discovery/req-topics-blacklisting`
LABEL	/conf/discovery/api-sta-websub-landing-page
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 GET request on each URL. 3. Validate the returned link headers to contain the `link rel=self"` and **not** `link rel="help"`.

A.1.23. Example root page snippet advertising no denied topics

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

Listing A.14

A.1.24. Example HTTP request and response for a URL not including a denied topic

```
GET /sta/v1.1/Observations HTTP/1.1
```

```
Host: example.com

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

Listing A.15

B

ANNEX B (INFORMATIVE) HUB IMPLEMENTATION GUIDANCE

B

ANNEX B (INFORMATIVE) HUB IMPLEMENTATION GUIDANCE

This appendix provides guidance as to how to implement a STA-WebSub Hub (implementation) for a SensorThings API 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 done because the STA-WebSub Hub is not the standardization target!

B.1. Callback Authentication

A Hub 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` or `hub.x_api_key` parameter SHOULD only be specified with static callback URLs. The parameter MUST be less than 200 bytes in length.

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

One fundamental (runtime) functionality for a WebSub Hub is the ability to deliver a notification content to all subscribers.

The W3C WebSub offers the use of HMAC so that subscribers can validate the authenticity of received content; Basically check that the content was received from the expected hub. Leveraging HMAC also enables the subscriber to validate content integrity which is important

when using non secure communication — so HTTP instead of HTTP**S**. The W3C WebSub supports the transmission of the HMAC using the HTTP header `X-Hub-Signature`.

However, the use of HMAC signatures however introduces a burden to the hub's and subscriber's CPU when calculating the HMAC value. This burden also requires caching of the content at the hub and at the subscriber implementation. The use of HMAC prevents the subscriber to stream the received content directly into connected workflow processing. However, on the hub side, the caching does not introduce a resource burden because an implementation would probably cache the content anyway before distributing it to all subscribers.

Because the validation of the HMAC requires that a subscriber receives the entire content, this introduces a denial-of-service vulnerability. Assuming that an attacker would send large fraudulent content with high frequency using many parallel requests, the subscriber must process the entire content to determine authenticity.

The W3C WebSub recommends that a callback URL is random and gets refreshed by the subscriber when a subscription is updated. This procedure gains equivalent protection of the callback endpoint to the use of API key. The use of HMAC is not necessary, assuming that a callback endpoint is associated with one hub only and the callback operates over HTTPS.

However, when the generation or refreshing of callback URLs is not possible or too difficult, STA-WebSub offers the use of API key authentication. The use of API key compensates for the use of random callback URLs. Further, API key in combination with HTTP**S** callback URLs is equivalent to the use of HMAC in terms of authenticity and integrity but it is not necessary to calculate HMAC on the hub side and validate on the subscriber side. This reduces the burden on resources at the hub and subscriber side. This is important for hub deployments on the edge.

Like the use of `hub.secret` to share the secret for HMAC generation, a STA-WebSub subscription may include the parameter `hub.api_key` or `hub.x_api_key` to trigger API key management at the hub. These parameters trigger that the hub adds the HTTP header `Api-Key` or `X-Api-Key` when distributing the content to the subscriber.

B.3. Subscription

The STA-WebSub Hub, associated with a SensorThings API service, supports the W3C subscribe / unsubscribe protocol and transforms a subscription topic URL (`hub.topic`) into a MQTT topic. A compliant implementation must know how to transform the absolute HTTP(S) URL into the corresponding MQTT topic for the associated SensorThings API service.

The Hub uses the MQTT protocol to subscribe and unsubscribe with the MQTT broker of the associated SensorThings API service.

An implementation of the STA-WebSub Hub MUST transform the HTTP (discovery) URL into a MQTT topic by removing the SensorThings API `baseUrl`. For example, a deployment with the `baseUrl=http://localhost:8080/mysta`, the discovery URL `http://localhost:8080/mysta/v1.1/Observations` results in the MQTT topic `v1.1/Observations`.

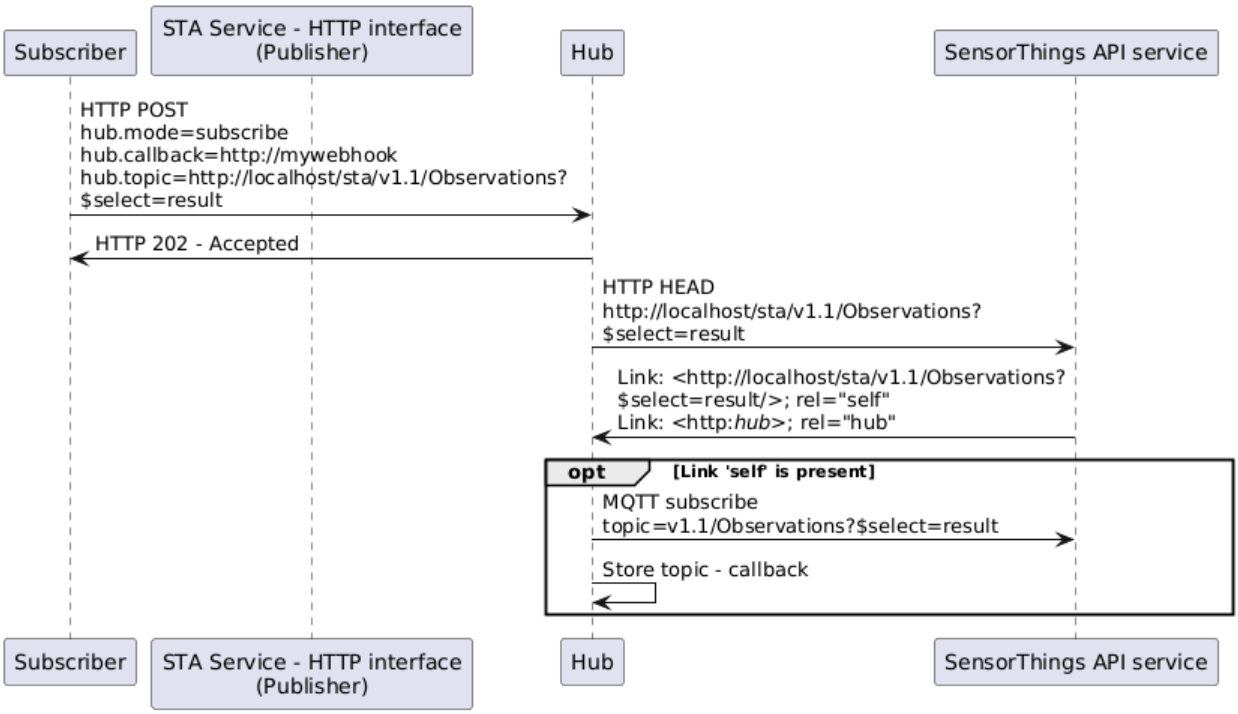


Figure B.1 — Subscription Handling in the STA-WebSub Hub

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, the Hub SHOULD validate the intent of the subscriber as defined in the W3C WebSub. For a STA-WebSub Hub checking the subscription validity with the publisher using the discovery protocol is also recommended. Upon success (the `Link rel="self"` is included in the discovery response), the Hub subscribes to the SensorThings API service via MQTT after transforming the topic URL into a MQTT topic.

Using the HTTP HEAD method is recommended for the discovery requests to avoid unnecessary data transfer.

For a new or renewing subscription (the hub receives a subscribe request for an existing subscription), the implementation MUST set/update the API key and use the value for further 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. An implementation MUST listen to notifications from the MQTT broker of the SensorThings API service. Once a notification is received, the

implementation MUST distribute the notification content to all subscribers using HTTP(S) as defined by the W3C WebSub Recommendation.

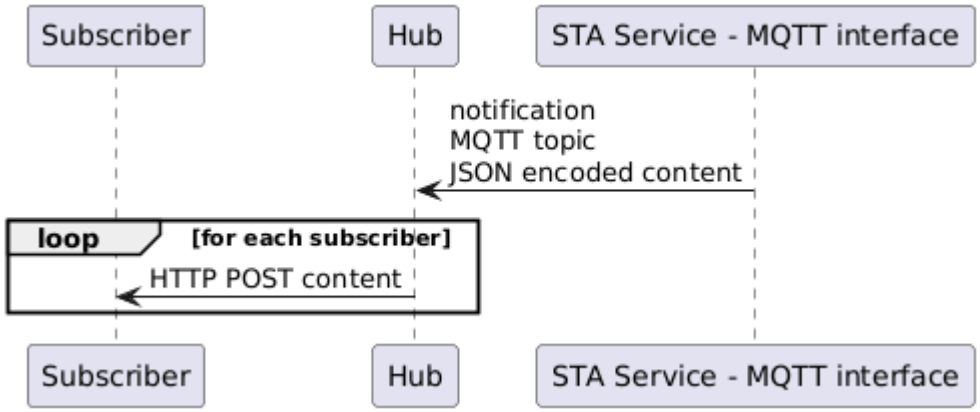


Figure B.2 — MQTT notification delivery by the STA-WebSub Hub

If the subscription was made with the `hub.api_key` or `hub.x_api_key`, the implementation MUST add the `Api-Key` or `X-Api-Key` to the HTTP request send to the subscriber's callback.

Likewise, the implementation MUST add the `X-Hub-Signature` header if the subscription was made with the `hub.secret` parameter.

B.5. Implementations

An implementation of a STA-WebSub system is available as open source:

- The STA-WebSub Hub is available as open source from Github: https://github.com/securedimensions/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
  - 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/WebSub-Subscriber

C

ANNEX C (INFORMATIVE) REVISION HISTORY

C

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 corrects from Carl Reed

BIBLIOGRAPHY

BIBLIOGRAPHY