OGC® DOCUMENT: 23-013

External identifier of this OGC® document: http://www.opengis.net/doc/discussion-paper/ogcapi-pubsub/1.0

Open Geospatial Consortium

DISCUSSION PAPER FOR PUBLISH-SUBSCRIBE WORKFLOW IN OGC APIS

DISCUSSION PAPER

PUBLISHED

Submission Date: 2023-02-02
Approval Date: 2023-06-08
Publication Date: 2023-10-26
Editor: Tom Kralidis, Mark Burgoyne, Steve Olson, Shane Mill

Note

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

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

CONTENTS

LIST OF TABLES

LIST OF FIGURES

I ABSTRACT

OGC APIs provide Web based capabilities which are typically based on polling for collection resource updates (new features/records items, coverages, maps, etc.). Depending on a collection's temporal resolution or frequency of updates, an event-driven / Publish-Subscribe architecture provides a timely, efficient, and low latency approach for delivery of data updates. This paper provides recommendations on applying Publish-Subscribe architectural patterns to OGC APIs.

II KEYWORDS

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

OGC API, Pub/Sub, Publish, Subscribe, Publish-Subscribe, Event driven architecture, Asynchronous, OGC document, OGC

III PREFACE

This document provides an overview of Publish-Subscribe patterns specific to event driven data workflows, as well as options for realizing Publish-Subscribe workflows in OGC APIs. The document builds on the OGC Publish-Subscribe White Paper (OGC document 20-081) and provides recommendations for baseline standardization within the OGC API ecosystem.

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

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

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

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

- Meteorological Service of Canada
- UK Met Office
- US National Weather Service

VI SUBMITTERS

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

NAME	AFFILIATION
Steve Olson	NOAA/NWS
Tom Kralidis	Meteorological Service of Canada
Mark Burgoyne	UK Met Office
Shane Mill	NOAA/NWS

1

SCOPE

1 SCOPE

This Discussion Paper provides insight into the ongoing work within the OGC MetOcean Domain Working Group (MetOceanDWG) and related activities (World Meteorological Organization Information System version 2 [WIS2])) regarding Publish-Subscribe architecture. This paper makes a case for a Publish-Subscribe standard in the OGC API ecosystem.

One of the cases the authors of this paper would like to make in the OGC API — EDR Pub/Sub work is the broader appeal for the larger set of OGC API's. The basis for this is the author's choice for the Pub/Sub channel structure to match the OGC API "collections" structure. The overall benefit is that the building blocks of the RESTful API endpoint structure would be crosswalked as Pub/Sub channels with the only difference in syntax being dot notation for channels rather than "/" notation. This allows for a parallel access pattern between the RESTful API and Pub/Sub access.

2

NORMATIVE REFERENCES

2 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.

IANA Link Relations https://www.iana.org/assignments/link-relations/link-relations.xhtml

Aaron Braeckel, Lorenzo Bigagli, Johannes Echterhoff: OGC 13-131r1, *OGC® Publish/Subscribe Interface Standard 1.0 — Core*. Open Geospatial Consortium (2016). https://docs.ogc.org/is/13-131r1/13-131r1.html.

OGC Publish-Subscribe White Paper (2020)

Mark Burgoyne, David Blodgett, Charles Heazel, Chris Little: OGC 19-086r5, *OGC API — Environmental Data Retrieval Standard*. Open Geospatial Consortium (2022). https://docs.ogc.org/is/19-086r5/19-086r5.html.

Clemens Portele, Panagiotis (Peter) A. Vretanos, Charles Heazel: OGC 17-069r4, *OGC API — Features — Part 1: Core corrigendum*. Open Geospatial Consortium (2022). https://docs.ogc.org/is/17-069r4/17-069r4.html.

DRAFT OGC API — Records — Part 1: Core (2020)

H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub: IETF RFC 7946, *The GeoJSON Format*. RFC Publisher (2016). https://www.rfc-editor.org/info/rfc7946.

Advanced Message Queueing Protocol (AMQP) v1.0) https://www.oasis-open.org/standard/amqp

MQTT Version 3.1.1 Plus Errata 01 https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

MQTT Version 5.0 https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

AsyncAPI Specification https://www.asyncapi.com/docs/reference/specification/v2.5.0

Draft guidance on technical specifications of WIS 2.0 https://wmoomm.sharepoint.com/sites/wmocpdb/eve_activityarea/Forms/AllItems.aspx?id=%2Fsites%2Fwmocpdb%2Feve%5Factivityarea%2FWMO%20Information%20System%20%28WIS%29%5F6cf41395%2D0ffc%2Dea11%2Da813%2D000d3aafe55d%2Fdocs%20shared%20on%20community%20site%2FWIS%202%2E0%20Guidance%2FGuidance%20on%20technical%20specification%20of%20WIS2%2Dv2%2Epdf&parent=%2Fsites%2Fwmocpdb%2Feve%5Factivityarea%2FWMO%20Information%20System%20%28WIS%29%5F6cf41395%2D0ffc%2Dea11%2Da813%2D000d3aafe55d%2Fdocs%20shared%20on%20community%20site%2FWIS%202%2E0%20Guidance&p=true&ga=1

DRAFT WMO WIS2 Notification Message https://github.com/wmo-im/wis2-notification-message

DRAFT WMO WIS2 Topic Hierarchy https://github.com/wmo-im/wis2-topic-hierarchy

WebSockets https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

3

TERMS, DEFINITIONS AND ABBREVIATED TERMS

3 TERMS, DEFINITIONS AND ABBREVIATED TERMS

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

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

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

This document uses the terms defined in Sub-clause 5.3 of [OGC06-121r9], 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 standard.

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

3.1. Broker

Intermediary between Subscribers and other Publishers which have been previously registered with the Broker. The Broker is not the original producer of Messages, but acts as an intermediary, (re-)publishing messages received from other Publishers and decoupling them from their Subscribers.

3.2. Subscriber

An entity that creates a subscription to a Publisher.

3.3. Message

A container within which data (such as JSON, XML, binary data, or other content) is transported. Messages may include additional information beyond data, including headers or other metadata used for routing or security purposes.

3.4. Channel

A term (string) used to filter messages from a Broker.

3.5. Abbreviated terms

AMQP	Advanced Message Queuing Protocol
AMQPS	Advanced Message Queuing Protocol Secure
API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
GeoJSON	Geospatial JavaScript Object Notation (JSON)
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
JSON	JavaScript Object Notation
MQP	Message Queuing Protocol
MQTT	Message Queuing Telemetry Transport
MQTTS	Message Queuing Telemetry Transport Secure
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
URI	Uniform Resource Identifier

WIS	WMO Information System
WMO	World Meteorological Organization
YAML	YAML Ain't Markup Language

4

CONVENTIONS

4 CONVENTIONS

This section 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.

4.1. Identifiers

The normative provisions in this document are denoted by the URI

`http://www.opengis.net/spec/ogcapi-pubsub/1.0`

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

5

OVERVIEW

5 OVERVIEW

Event-driven workflows provide publish-subscribe based capabilities as part of information systems and architectures. The publish-subscribe model also provides efficiencies in providing data “as it happens,” thereby alleviating potential clients from continuous polling to check on the availability of new data/resources.

The Open Geospatial Consortium (OGC) has conducted significant work on event-based models and architectures. The publish-subscribe model results in less network traffic and more timely responses to manage event-based models such as urgent, temporally unpredictable data (examples include, but are not limited to weather or hazard warnings and sensor data).

Building on the OGC Publish-Subscribe Interface Standard, as well as the recommendations put forward in the OGC Pub/Sub White Paper (OGC 20-046) produced as part of OGC Testbed 12, this paper discusses approaches for integrating publish-subscribe architecture into the OGC API suite of standards.

6

OGC API CONSIDERATIONS

6 OGC API CONSIDERATIONS

The OGC API building block approach is typically implemented for shared components of APIs in support of polling workflows. This means the client initiates and invokes requests and receives responses from the server using HTTP. A key concept of the OGC API building blocks is the endpoint hierarchy, which can be applied for Pub/Sub workflow as follows:

- data producers: messages are published to a broker, applied to a given channel (example: `collections/mycollection`);
- broker provisioning: published messages are sent to subscribers; and
- subscribers and data consumers: messages are received by users subscribed to one or more channels (explicitly or using wildcards/filtering).

The above workflow requires adherence to a hierarchy of channels and autodiscovery of channels, as well as processing of generic messages for broad interoperability by all components.

6.1. Channels, message payloads and OGC API endpoints

The OGC API endpoint hierarchy can be used in parallel as a channel description when the data publisher wishes to provide Pub/Sub capability for OGC API resources normally available in the same way. Below are examples of OGC endpoints normally available via HTTP, and how these endpoints can be re-used as topics for Pub/Sub workflow as follows.

- `/collections`: notifies Subscribers whenever there is a change to the `/collections` OGC API endpoint (for example, addition of a new collection). The message payload would be collection metadata as defined by OGC API — Common, or a message referencing the same.
- `/collections/{collectionId}`: notifies Subscribers whenever there is an update to a single collection (for example, spatial or temporal extents, new items, etc.). The message payload would be defined by the resource model of the given collection (items, etc.), or a message referencing the same.

For smaller payload workflows, OGC APIs can additionally provide a channel for notification metadata in order to receive a smaller message payload, which a user can process to determine whether to further interact with the reference data granule or resource.

Using the OGC API endpoint hierarchy provides the key benefit that OGC API users do not need to learn a different, additional approach or hierarchy for Pub/Sub (same content, additional interface).

6.2. AsyncAPI

Following the recommendations of the Pub/Sub White Paper, AsyncAPI provides an event-driven equivalent of what is provided by OpenAPI for OGC APIs (description of protocols, channels, parameters, models, etc.). An OGC API landing page can provide a link to an AsyncAPI document as follows:

```
{
    "rel": "http://www.opengis.net/def/rel/ogc/1.0/pubsub/asyncapi",
    "type": "application/json",
    "title": "AsyncAPI document",
    "href": "https://example.org/asyncapi"
}
```

Figure 1 — OGC API landing page AsyncAPI link example

6.3. Providing notification metadata as an OGC API endpoint

For Brokers providing notification metadata (as opposed to actual data payloads), an OGC API can, in parallel, easily provide GeoJSON-based notification messages via an OGC API — Features endpoint. Providing message payloads via an OGC API provides the additional benefit of querying for past messages over time in case of a lost connection.

6.4. Providing Pub/Sub links to collection updates

The links array could also provide references to the Pub/Sub capabilities available on the service; a **collection** link could reference a collection update notification channel:

```
{
    "rel": "collection",
    "type": "application/json",
    "title": "Data notifications",
    "href": "mqtts://example.org",
    "channel": "collections"
}
```

Figure 2 — OGC API Pub/Sub link example to new collection notifications

An **items** link could reference a data payload channel:

OGC API — Features example

```
{
    "rel": "items",
```

```
    "type": "application/json",
    "title": "Data notifications",
    "href": "mqtts://example.org",
    "channel": "collections/{collectionId}"
}
```

Figure 3

OGC API — EDR example

```
{
    "rel": "items",
    "type": "application/json",
    "title": "Data notifications",
    "href": "mqtts://example.org",
    "channel": "collections/{collectionId}/items"
}
```

Figure 4

7

RECOMMENDATIONS FOR AN OGC API – PUB/SUB STANDARD

7 RECOMMENDATIONS FOR AN OGC API — PUB/SUB STANDARD

An OGC API — Pub/Sub standard can provide the following requirements, recommendations, and permissions for server implementations.

7.1. API conformance

It is appropriate to add a conformance class to describe that the service supports Pub/Sub with reference to the specifications needed to define the functionality. The requirements class and the conformance class with a set of conformance tests covering a variety of protocol options that ensure compliance with the standard must also be defined.

An OGC API server SHALL provide a mechanism to define the protocols and message schemas it supports following the convention in the use of OpenAPI to describe OGC API's, as well as the `/conformance` endpoint. An API SHALL NOT mandate any encoding or format for the formal definition of the API. This approach has allowed OGC API's to take advantage of industry best practices for describing functionality. At the time of this writing, AsyncAPI can be defined as the preferred option.

7.2. API definition

A link to the Pub/Sub definition document for the service SHALL be provided on the service landing page `/` as a link with a defined link relation as per IANA[1] or OGC API Standards.

7.3. Protocols and interoperability

An OGC API — Pub/Sub standard SHOULD NOT prescribe a protocol. An asynchronous API definition (such as AsyncAPI or equivalent technology), SHOULD be utilized to describe the interfaces supported by a service.

[1] https://www.iana.org/assignments/link-relations/link-relations.xhtml

7.4. Message payloads

An OGC API — Pub/Sub server SHOULD NOT mandate a message payload, but SHALL define and advertise all payload formats/schemas as part of the API definition. Examples include (but are not limited to) GeoJSON and various OGC API response schemas of metadata or actual data payloads or references to the same.

7.5. Channel descriptions

Channel (topic/destination/node depending on protocol) identifiers SHALL be based on the same naming hierarchy as the OGC API endpoints which are generating the events that users subscribe to.

8

WMO WIS2 IMPLEMENTATION

8 WMO WIS2 IMPLEMENTATION

The WMO Information System 2.0 (WIS 2.0) is the framework for WMO data sharing in the 21st century for all WMO domains and disciplines. It supports the WMO Unified Data policy, the Global Basic Observing Network (GBON) and makes international, regional, and national data sharing simple, effective, and inexpensive. The idea that no WMO Member should be left behind and the objective of lowering the barrier to adoption has been at the core of WIS 2.0 development. These objectives inspire the principles underpinning the WIS 2.0 technical framework, such as adopting open standards and Web technologies to facilitate sharing of increasing variety and volume of real-time data[2].

Given the real-time nature of weather/climate/water data, WIS2 makes significant use of Pub/Sub standards and technologies, and has resulted in the following draft standards:

- WIS2 Notification Message[3]
- WIS2 Topic Hierarchy[4]

Key core protocols and encodings of WIS2 are identified below:

- HTTP
- MQTT
- GeoJSON

A draft architecture is illustrated below, exemplifying the Pub/Sub interactions between WIS2 Nodes and the Global Broker (a WIS2 global service).

[2] https://community.wmo.int/activity-areas/wis

[3] https://github.com/wmo-im/wis2-notification-message

[4] https://github.com/wmo-im/wis2-topic-hierarchy

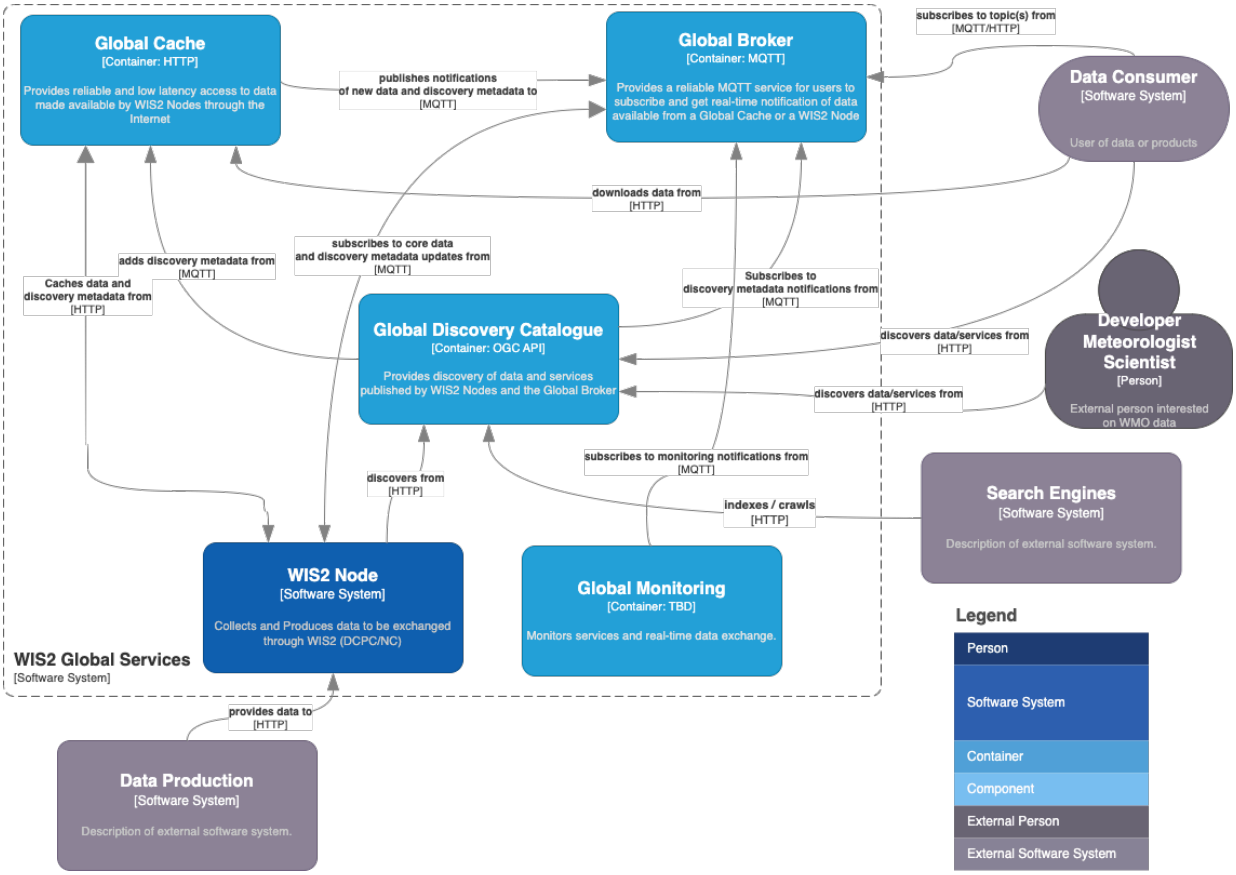

Figure 5

8.1. Protocols and Message Queuing Protocol (MQP) Brokers

WIS2 mandates use of a Message Queuing Protocol (MQP) using MQTT 3.1.1 and/or MQTT 5.0. MQTT can also be used over the Web using WebSockets, thereby providing the ability to build mobile and web applications.

8.2. Notification Message

The WIS2 standard notification message format ensures that the WIS2 ecosystem (data publisher, data user, and global services) is a robust, effective, and unified exchange platform for weather, climate, and water data. The message provides notification metadata about the availability of a new data granule. The message is encoded using a GeoJSON baseline and provides detailed information on the data notification (associated datetime of the granule, publishing datetime, integrity), as well as access to the data via a link object or inline content

(useful for encoding small messages). Geometry is required (given GeoJSON requirements) but can be expressed with a `null` value when generating the geometry in the message is not possible, practical, or timely for data publishers. To support extensibility, additional properties are also valid (given the default definition in JSON Schema).

Using a GeoJSON baseline as the message payload allows for broad interoperability given the large ecosystem of tooling (decoders, encoders) supporting the same approach. An example web application demonstrating the ease of integration can be found at https://kralidis.ca/tmp/wis2-data-notifications.html.


```
{
    "id": "31e9d66a-cd83-4174-9429-b932f1abe1be",
    "version": "v04",
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [
            6.146255135536194,
            46.223296618227444
        ]
    },
    "properties": {
        "pubtime": "2022-03-20T04:50:18.314854383Z",
        "datetime": "2022-03-20T04:45:00Z",
        "integrity": {
            "method": "sha512",
            "value": "A2KNxvks...S8qfSCw=="
        },
        "data_id": "wis2/CAN/eccc-msc/data/core/weather/surface-based-obs/
landFixed/UANT01_CWAO_200445___15103.bufr4",
        "content": {
            "encoding": "utf-8",
            "value": "encoded bytes from the file",
            "size": 457
        }
    },
    "links": [
        {
            "href": "https://example.org/data/4Pubsub/92c557ef-d28e-4713-91af-
2e2e7be6f8ab.bufr4",
            "rel": "canonical",
            "type": "application/x-bufr"
        }
    ]
}
```


Figure 6 — Example of a WIS2 Notification Message

8.3. Topic hierarchy

The WIS2 topic hierarchy provides a central classification and categorization scheme used by data providers and WIS2 Global Services in support of core WIS2 workflows: publish, discover, subscribe, and download, and is aligned with and implements WMO Res. 1 Cg-EXT-21 — Unified

Data Policy[5]. The topic hierarchy defines eight (8) core levels for all WIS2 data and provides an extensibility framework for specific information communities (atmospheric composition, cryosphere, etc.) to define their own hierarchy as subdomains.

Combined with implementing MQP brokers, the topic hierarchy allows for subscribing and filtering of notification messages by data consumers.

[5] https://ane4bf-datap1.s3-eu-west-1.amazonaws.com/wmocms/s3fs-public/ckeditor/files/Cg-Ext2021-d04-1-WMO-UNIFIED-POLICY-FOR-THE-INTERNATIONAL-approved_en_0.pdf?4pv38FtU6R4fDNtwqOxjBCndLIfntWeR

A

ANNEX A (INFORMATIVE) EXAMPLES (INFORMATIVE)

A ANNEX A (INFORMATIVE) EXAMPLES (INFORMATIVE)

A.1. Pub/Sub API Description Example

The API is described using the AysncAPI 2.5.0 specification and an example response can be found below:

Example 1

```
asyncapi: '2.5.0'
info:
    title: AsyncAPI demo
    version: '0.0.1'
    description: |
        AsyncAPI description of the proposed Pub/Sub functionality
    contact:
        name: Contact Name
        email: you@example.org

servers:
    mqtt_prod:
        url: example.org
        protocol: mqtt
        description: MQTT endpoint
        security:
            - user-password: []
defaultContentType: application/json

channels:
    collections:
        subscribe:
            message:
                $ref: '#/components/messages/collection_msg'
    collections/wthr_stn:
        subscribe:
            message:
                $ref: '#/components/messages/collection_msg'
    collections/stream_gage:
        subscribe:
            message:
                $ref: '#/components/messages/collection_msg'
    collections/wthr_stn/items:
        subscribe:
            message:
                $ref: '#/components/messages/wthr_stn_msg'
    collections/stream_gage/items:
```

```
        subscribe:
            message:
                $ref: '#/components/messages/stream_gage_msg'

components:
    messages:
        collection_msg:
            description: collection updated notification
            payload:
                type: object
                required:
                    - id
                    - href
                properties:
                    id:
                        type: string
                        description: collection name
                    time:
                        type: string
                        format: date-time
                        description: time collection changed
                    href:
                        type: string
                        format: uri
                        description: URL of the changed collection
        wthr_stn_msg:
            description: An observation formatted as GeoJSON
            payload:
                type: object
                additionalProperties: false
                properties:
                    id:
                        type: string
                    type:
                        type: string
                    geometry:
                        type: object
                        properties:
                            type:
                                type: string
                            coordinates:
                                type: array
                                items:
                                type: number
                                format: float
                    properties:
                        type: object
                        properties:
                            time:
                                type: string
                                format: date-time
                            id:
                                type: string
                            wind_direction:
                                type: number
                                format: float
                            wind_speed:
                                type: number
                                format: float
                            wind_gust:
                                type: number
                                format: float
                            visibility:
```

```
                                type: number
                                format: float
                            air_temperature:
                                type: number
                                format: float
                            dew_point:
                                type: number
                                format: float
                            mean_sea_level_pressure:
                                type: number
                                format: float
        stream_gage_msg:
            description: Monitoring station data formatted as GeoJSON
            payload:
                type: object
                additionalProperties: false
                properties:
                    id:
                        type: string
                    type:
                        type: string
                    geometry:
                        type: object
                        properties:
                            type:
                                type: string
                            coordinates:
                                type: array
                                items:
                                type: number
                                format: float
                links:
                    type: array
                        items:
                            type: object
                            properties:
                                rel:
                                    type: string
                                type:
                                   type: string
                                title:
                                   type: string
                                href:
                                    type: string
                                format: uri
                properties:
                    type: object
                    properties:
                        datetime:
                            type: string
                            format: date-time
                        label:
                            type: string
                        parametername:
                            type: array
                            items:
                                type: string
                        edrqueryendpoint:
                            type: string
                            format: uri
securitySchemes:
    user-password:
        type: userPassword
```

Breaking down into the components:

Example 2

```
asyncapi: '2.5.0'
info:
    title: AsyncAPI demo
    version: '0.0.1'
    description: |
        AsyncAPI description of the proposed Pub/Sub functionality
    contact:
        name: Contact Name
        email: you@example.org
```

- The `asyncapi` field indicates the user should use the AsyncAPI version 2.5.0.
- The `info` field holds information about the API, such as its name, version, description, and license.

Example 3

```
servers:
    mqtt_prod:
        url: example.org
        protocol: mqtt
        protocolVersion: 3.1.1
        description: MQTT endpoint
        security:
            - user-password: []
```

- Each `server` object provides the following fields:
    - `url`: URL of the target broker (this may be relative to the API document);
    - `protocol`: Pub/Sub protocol supported by the server;
    - `protocolVersion`: version of the Pub/Sub protocol supported by the server;
    - `description`: string describing the host; and
    - `security`: reference to supported authentication types.

Example 4

```
channels:
    collections:
        subscribe:
            message:
                $ref: '#/components/messages/collection_msg'
    collections/wthr_stn:
        subscribe:
            message:
                $ref: '#/components/messages/collection_msg'
    collections/stream_gage:
        subscribe:
            message:
                $ref: '#/components/messages/collection_msg'
```

```
collections/wthr_stn/items:
    subscribe:
        message:
            $ref: '#/components/messages/wthr_stn_msg'
collections/stream_gage/items:
    subscribe:
        message:
            $ref: '#/components/messages/stream_gage_msg'
```

- The `channels` section lists the events a user can subscribe to and can provide a schema for the associated message payloads.
- In the example the following events can be subscribed to:
  - `collections`
  - `collections/wthr_stn`
  - `collections/stream_gage`
  - `collections/wthr_stn/items`
  - `collections/stream_gage/items`

Example 5

```
components:
```

- As in the OpenAPI specification, the `components` section is used to define reusable objects for different aspects of the AsyncAPI specification.

B

ANNEX B (INFORMATIVE) REVISION HISTORY

B

ANNEX B (INFORMATIVE) REVISION HISTORY

Table B.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2022-11-22	0.1	S. Olson	all	initial version
2023-01-15	0.2	T. Kralidis	all	bootstrap
2023-01-18	0.2.1	C. Little	all	editorial changes
2023-01-19	0.2.2	M. Burgoyne	all	editorial changes
2023-01-21	0.2.3	S. Olson/S. Mill	all	editorial changes
2023-01-25	0.3	M. Burgoyne	all	added annex
2023-01-25	0.3	M. Burgoyne	all	added annex
2023-01-26	0.3	T. Kralidis	all	updates based on MetOceanDWG 2022-01-26 meeting

BIBLIOGRAPHY

BIBLIOGRAPHY

NOTE: The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

[1] *OGC: OGC Testbed 12 Annex B: Architecture* (2015).