# WMO WIS2 Notification Message Encoding

# World Meteorological Organization

Date: 2023-03-06

Version: 1.0.0-DRAFT-2023-03-06

Document location: TBD

Document status: DRAFT

Standing Committee on Information Management and Technology (SC-IMT)[1]

Commission for Observation, Infrastructure and Information Systems (INFCOM)[2]

# Table of Contents

## i. Abstract

WIS2 real-time data sharing is based on a message queuing protocol (MQP) supporting a publication/subscription (PubSub) mechanism. A user can subscribe to an MQP broker to receive real-time notifications of the existence of new data. The notification message received from the MQP broker contains a URL to download the data. In addition, the MQP broker offers a range of topics organised in a hierarchy. The users can select their topics of interest and subscribe to them to receive notifications and download data relevant to their work.

The 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.

This document defines the content, structure, and encoding for the WIS2 Notification Message Encoding. This standard is an extension of the OGC API - Features standard [3]. WIS2 Notification Message documents are provided as MQP payloads by WIS2 nodes, Global Broker services, as well as Replay API services (optional OGC API - Features services for data notifications).

WIS2 Notification Message documents shall be encoded in GeoJSON (RFC7946 [4]) as defined in this specification and shall be made available as MQP payloads. Additionally, they can be provisioned as defined by OGC API - Features.

Weather/climate/water data is by nature geospatial and temporal. The W3C Data on the Web Best Practices [5] and Spatial Data on the Web Best Practices [6] publications provide guidelines on how to best enable spatiotemporal data to lower the barrier for users, search engine optimization, and linked data. This also aligns with the FAIR data principles (Findable, Accessible, Interoperable, Reusable) [7].

## ii. Keywords

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

wmo, wis 2.0, weather, climate, water, metadata, pubsub, mqp, message queuing protocol, GeoJSON

## iii. Security Considerations

Based on the WMO Unified Data Policy for the International Exchange of Earth System Data (Resolution 1 (Cg-Ext(2021) [8], exchanged data are classified as `core` or `recommended`. Core data is considered fully open and unrestricted with no security considerations. Recommended data may have access control defined.

No security considerations have been made for this standard.

# Chapter 1. Scope

This document defines the content, structure, and encoding of a notification message published as part of a WIS2 Global Broker or a Replay API service.

The WIS2 Notification Message Encoding standard defined herein is an extension of the International Standard *GeoJSON*.

WIS2 Notification Message documents shall be encoded as GeoJSON as defined in *OGC API - Features - Part 1: Core*.

The primary purpose of WIS2 Notification Messages are to notify subscribers or clients of new data being published.

This specification defines the conformance requirements for the WIS2 Notification Message Encoding. Annex A defines the abstract test suite.

[1] https://community.wmo.int/governance/commission-membership/commission-observation-infrastructures-and-information-systems-infcom/commission-infrastructure-officers/infcom-management-group/standing-committee-information-management-and-technology-sc-imt

[2] https://community.wmo.int/governance/commission-membership/infcom

[3] https://ogcapi.ogc.org/features

[4] https://datatracker.ietf.org/doc/html/rfc7946

[5] https://www.w3.org/TR/dwbp

[6] https://www.w3.org/TR/sdw-bp

[7] https://en.wikipedia.org/wiki/FAIR_data

[8] https://library.wmo.int/doc_num.php?explnum_id=11113#page=9

# Chapter 2. Conformance

Conformance with this standard shall be checked using the tests specified in Annex A (normative) of this document.

*GeoJSON* provides an encoding for encoding geographic features (geometry, attributes). This standard is an extension of *GeoJSON*.

Data providers are required to comply with all conformance classes of this specification in support of providing MQP services.

WMO shall publish guidance material to assist data providers in constructing WIS2 Notification Messages.

This standard identifies one Conformance Class which defines the functional requirements.

The mandatory Conformance Class for this specification is:

- "WIS2 Notification Message Encoding Core"

# Chapter 3. References

- OGC: OGC 17-069r, OGC API - Features - Part 1: Core 1.0 (2022) [9]
- IETF: RFC-7946 The GeoJSON Format (2016) [10]
- IETF: RFC-8259 The JavaScript Object Notation (JSON) Data Interchange Formathe GeoJSON Format (2016) [11]
- W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note (2017) [12]
- W3C: Data on the Web Best Practices, W3C Recommendation (2017) [13]
- W3C: Data Catalog Vocabulary, W3C Recommendation (2014) [14]
- IANA: Link Relation Types (2020) [15]
- IETF: JSON Schema (2022) [16]
- WMO: WIS2 Topic Hierarchy (2022) [17]

[9] https://docs.opengeospatial.org/is/17-069r4/17-069r4.html

[10] https://datatracker.ietf.org/doc/html/rfc7946

[11] https://datatracker.ietf.org/doc/html/rfc8259

[12] https://www.w3.org/TR/sdw-bp

[13] https://www.w3.org/TR/dwbp

[14] https://www.w3.org/TR/vocab-dcat

[15] https://www.iana.org/assignments/link-relations/link-relations.xml

[16] https://json-schema.org

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

# Chapter 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 Standard 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.

The following additional terms and definitions also apply.

## 4.1. Abbreviated terms

*Table 1. Symbols and abbreviated terms*

| Abbreviation | Term |
|---|---|
| API | Application Programming Interface |
| DCPC | Data Collection and Production Centres |
| GDC | Global Discovery Catalogue |
| GIS | Geographic Information System |
| GISC | Global Information System Centre |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| IANA | Internet Assigned Numbers Authority |
| IETF | Internet Engineering Task Force |
| ISO | International Organization for Standardization |
| JSON | JavaScript Object Notation |
| MQP | Message Queuing Protocol |
| MQTT | Message Queuing Telemetry Transport |
| NC | National Centre |
| NWP | Numerical Weather Prediction |
| OGC | Open Geospatial Consortium |
| PubSub | Publish / Subscribe |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| UUID | Universally Unique Identifier |

| Abbreviation | Term |
| --- | --- |
| W3C | World Wide Web Consortium |
| WCMP | WMO Core Metadata Profile |
| WIS | WMO Information System |
| WMO | World Meteorological Organization |
| WNM | WIS2 notification message |

# Chapter 5. Conventions

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of JSON 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://wis.wmo.int/spec/wnm/1.0

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

## 5.2. Examples

Examples provided in this specification are encoded as GeoJSON.

Complete examples can be found at https://schemas.wmo.int/wnm/1.0/examples

## 5.3. Schemas

The WIS2 Notification Message schema can be found at https://schemas.wmo.int/wnm/1.0/notificationMessageGeoJSON.yaml

## 5.4. Schema representation

JSON Schema [18] objects are used throughout this standard to define the structure of metadata records. These schema objects are also typically represented using YAML [19]. YAML is a superset of JSON, and in this standard are regarded as equivalent.

Metadata record instances are always defined as JSON.

### 5.4.1. Properties

A JSON **property** represents a key-value pair, where the key is the name of the property and the value is a standard JSON data type.

```
"myPropertyName": "test123"
```

[18] https://json-schema.org

[19] https://en.wikipedia.org/wiki/YAML

# Chapter 6. Introduction

## 6.1. Motivation

MQP brokers provide "push" based services, supporting event-driven workflows, maximizing efficient use of network, bandwidth, and rapid response to time-sensitive events (e.g. severe weather events). However, MQP brokers do not specify a payload encoding. Using GeoJSON as a baseline for this specification provides broad interoperability, lowering the barrier and extending the reach of data in the WIS2 ecosystem and beyond.

## 6.2. Scenarios

The following scenarios are useful in understanding the drivers and principles that were used in the development of this specification:

### 6.2.1. Publish, Subscribe, Download

Event driven (PubSub) workflow involves a client connecting to a MQP broker, subscribing to one or more topics and receiving relevant notifications. Notifications provide information to ensure data integrity and spatiotemporal extents of a data granule.

### 6.2.2. Replay API workflow

API workflow involves a client connecting to an OGC API - Features services in order to query for messages in the past using spatial, temporal, or attribute predicates. A Replay API workflow is valuable in situations where connections to MQP brokers may drop, thereby allowing a given subscriber to gather past messages.

# Chapter 7. The WIS2 Notification Message Encoding

A WIS2 Notification Message provides descriptive information about a data made available through WIS 2.0.

## 7.1. Conformance Class Core

### 7.1.1. Overview

This Core Conformance Class provides requirements to articulate the required elements of a WIS2 Notification Message.

| **Requirements Class** | |
|---|---|
| http://www.wmo.int/spec/wnm/1.0/req/core | |
| Target type | Notification Metadata |
| Dependency | IETF RFC8259: The JavaScript Object Notation (JSON) Data Interchange Format |
| Dependency | JSON Schema |
| Dependency | GeoJSON |
| Pre-conditions | The record conforms to GeoJSON (RFC7946) |

The standard notification message format ensures that the WIS2 ecosystem (data publisher, data user, and global services) is a robust, effective, and unified data exchange platform for weather, climate, and water.

### 7.1.2. GeoJSON compliance

The WIS2 Notification Message schema is based on *GeoJSON* (RFC7946) and its associated information model. Compliant messages are therefore compliant with *GeoJSON*.

| **Requirement 1** | **/req/core/validation** |
|---|---|
| A | Each WNM SHALL validate without error against the WNM schema. |
| B | Each WNM SHALL provide `id`, `version`, `type`, `geometry` and `properties` properties for GeoJSON compliance. |
| C | Each WNM record `type` property SHALL be set to a fixed value of `Feature` for GeoJSON compliance. |

### 7.1.3. Identifier

A universally unique identifier of the message using the UUID standard (RFC4122). The identifier is generated by the originator of the message. It provides the anti-loop feature that is needed to ensure that the message will be seen once by all Global Brokers. It remains the same throughout the lifetime of the message in the WIS2 ecosystem.

The message identifier is set to a new value by Global Cache services when they publish their modified message for accessing the original core data from the cache. The data_id is retained to ensure traceability and consistency of the same resource.

```
"id": "31e9d66a-cd83-4174-9429-b932f1abe1be"
```

| **Requirement 2** | **/req/core/identifier** |
|---|---|
| A | A WNM SHALL have an identifier via the `id` property. |
| B | A WNM identifier SHALL be provided as a Universally Unique Identifier (UUID). |

### 7.1.4. Version

The `version` property provides the version of WNM that the message conforms to.

| **Requirement 3** | **/req/core/version** |
|---|---|
| A | A WNM SHALL provide information on version conformance via the `version` property. |
| B | A WNM's `version` property SHALL be fixed to `v04` for this version of the specification. |

### 7.1.5. Type

The `type` property is a fixed value denoting the record as a GeoJSON Feature.

| **Requirement 4** | **/req/core/type** |
|---|---|
| A | A WNM SHALL provide a `type` property. |
| B | A WNM's `type` property SHALL provide a fixed value of `Feature`. |

### 7.1.6. Geometry

RFC7946 defines 7 types of geometry: `Point`, `MultiPoint`, `LineString`, `MultiLineString`, `Polygon`, `MultiPolygon`, and `GeometryCollection`. It has been decided to restrict those types to only two: `Point` and `Polygon`. It must be noted that the geometry key is mandatory in GeoJSON but can be of type `null`.

| **Requirement 5** | **/req/core/geometry** |
|---|---|
| A | A WNM SHALL provide a `geometry` property. |
| B | A WNM's `geometry` property SHALL only provide one of a `Point` or `Polygon` geometry, or a `null` value when a geometry value is unknown or cannot be determined. |

*Example: Point*

```
{
  ...
  "geometry": {
    "type": "Point",
    "coordinates": [
      6.146255135536194,
      46.223296618227444
    ]
  }
  ...
}
```

*Example: Point with elevation*

```
{
  ...
  "geometry": {
    "type": "Point",
    "coordinates": [
      6.146255135536194,
      46.223296618227444,
      392
    ]
  }
  ...
}
```

*Example: Polygon*

```
{
  ...
  "geometry": {
    "type": "Polygon",
    "coordinates": [[
      [-7.75,40.43],
      [-7.75,78.46],
      [71.91,78.46],
      [71.91,40.43],
      [-7.75,40.43]
    ]]
  }
  ...
}
```

*Example: null*

```
{
  ...
  "geometry": null
  ...
}
```

## 7.1.7. Properties

### 7.1.7.1. pubtime

The `pubtime` property identifies the date/time of when the file was posted/published, in RFC3339 format, in the UTC timezone (`Z`). The publication date/time is critical for subscribers to prevent message loss in providing awareness of how far behind the publisher they may be).

```
"properties": {
  ...
  "pubtime": "2022-03-20T04:50:18.3148543837"
  ...
}
```

| **Requirement 6** | **/req/core/pubtime** |
|---|---|
| A | A WNM SHALL provide a `properties.pubtime` property. |
| B | A WNM's `properties.pubtime` property SHALL be provided in RFC3339 format. |

| | |
|---|---|
| C | A WNM's `properties.pubtime` property SHALL be provided in the UTC timezone. |

#### 7.1.7.2. data_id

The `data_id` property uniquely identifies the data. It is comprised of two parts:

1. the topic where the message will be published as defined in https://community.wmo.int/wis2-topic-hierarchy, removing both the channel (origin or cache) and the version of the topic hierarchy.
2. a unique identifier of the data. It can be the filename used by the originating center or a unique UUID (RFC4122) or a value chosen by the originating center as long as it is unique over a 1 week period

| **Requirement 7** | **/req/core/data_id** |
|---|---|
| A | A WNM SHALL provide a `properties.data_id` property. |
| B | A WNM's `properties.data_id` property SHALL be unique. |
| C | A WNM's `properties.data_id` property SHALL contain a valid WIS2 topic, without the channel and version. |

```
"properties": {
  ...
  "data_id": "wis2/can/eccc-msc/data/core/weather/surface-based-
obs/landFixed/UANT01_CWAO_200445___15103.bufr4"
  ...
}
```

#### 7.1.7.3. Temporal description (datetime, start_datetime, end_datetime)

The `datetime` property identifies the date and time of the data being published (e.g. date of measurement, for observation data), in RFC3339 format, in the UTC timezone (`Z`).

The `start_datetime` and `end_datetime` properties identify a temporal extent (e.g. start/end times for an NWP forecasting period), in RFC3339 format, in the UTC timezone (`Z`).

| **Requirement 8** | **/req/core/temporal** |
|---|---|
| A | A WNM SHALL provide a temporal description by either a `properties.datetime` property or both the `properties.start_datetime` and `properties.end_datetime` properties. |

| | |
|---|---|
| B | A WNM's temporal description SHALL be provided in RFC3339 format. |
| C | A WNM's temporal description SHALL be provided in the UTC timezone. |

*Example: Temporal instant*

```
"properties": {
  ...
  "datetime": "2022-03-20T04:45:00Z"
  ...
}
```

*Example: Temporal extent*

```
"properties": {
  ...
  "start_datetime": "2022-03-20T04:45:00Z",
  "end_datetime": "2022-03-22T04:45:00Z"
  ...
}
```

#### 7.1.7.4. Integrity

For data verification, it is suggested (but not mandatory) to include data integrity information via the `integrity` property. Providing this information will allow data consumers to ensure that a given data granule has not been corrupted during download.

The `method` property provides a format of the hashing method used to enable integrity check of the data. Acceptable values are `md5`, `sha256` and `sha512`. `sha512` is preferred.

The `value` property provides the result of the hashing method.

| **Recommendation 1** | **/rec/core/integrity** |
|---|---|
| A | A WNM SHOULD provide an `integrity` property, consisting of a `method` property identifying the hashing method (`md5`, `sha256`, `sha512`) and a `value` property of the hashing result, when it can be easily derived. |

```
"properties": {
  ...
  "integrity": {
    "method": "sha512",
    "value": "3bb12eda3c298db5de25597f54d924f2e17e78a26ad89...b0124ecb8a"
  }
  ...
}
```

#### 7.1.7.5. Content

For data granules with sizes smaller than 2048 bytes, the `content` property allows for including the data in the message. The `content` property provides an additional inline data access capability, in addition to a canonical link of the message.

The `encoding` property provides the character encoding of the data (fixed to UTF-8 or Base64).

The `value` property provides the data in the in accordance to the `encoding` property.

The `size` property provides the size of the data in accordance to the `encoding` property. The value must be below 2048. Global Brokers may discard messages for where inline data sizes are greater than 2048 bytes.

| **Recommendation 2** | **/rec/core/content** |
|---|---|
| A | A WNM SHOULD provide a `content` property, consisting of an `encoding` property (fixed to either `utf-8`, `base64`, or `gzip`), a `value` property (of less than 2048 bytes) of the data, as well as a `size` property with the length of the data. |

```
"properties": {
  ...
  "content": {
    "encoding": "utf-8",
    "value": "encoded bytes from the file",
    "size": 457
  }
  ...
}
```

#### 7.1.7.6. Additional properties

A WIS2 Notification Message can be extended as required for organizational purposes by adding properties (of any type) in the message. Additional properties do not break compliance to this specification.

| Permission 1 | /per/core/additional_properties |
|---|---|
| A | A WNM MAY provide additional properties of any type in any part of the document as needed. |

```
"properties": {
  ...
  "_comment": {
    "validationErrors": [
      "error 1",
      "error 2"
    ]
  }
  ...
}
```

### 7.1.8. Links

The `links` array consists of one or more objects providing URLs to access data.

Each link object provides:

- an `href` property with a fully qualified link to access the data in a secure manner
- a `rel` property providing an IANA link relation [20] describing the relationship between the link and the message
- a `type` property providing the media type of the data
- a `length` property providing the length (in bytes) indicating the size of the data in the response when downloading the link.

| Requirement 9 | /req/core/links |
|---|---|
| A | A WNM SHALL define a `links` array property. |
| B | A WNM's `links` array property SHALL contain at least one link with at least the required `href` and `rel` properties. |
| C | A WNM's link object's `rel` property SHALL use one of valid IANA link relations [22]. |
| D | A WNM's `links` array property SHALL contain exactly one link with a link relation of `canonical` to clearly identify the preferred link from which to access data. |

| | |
|---|---|
| E | A WNM's `links` array property SHALL contain links which, for core data, require no further action in order to download a given data granule. |
| F | A WNM SHALL provide links using HTTP, HTTPS, FTP or SFTP. |

| **Recommendation 3** | **/rec/core/links** |
|---|---|
| A | A WNM SHOULD provide links using secure protocols such as HTTPS and SFTP, with HTTPS being the preferred option. |
| B | A WNM link SHOULD provide the `length` property to communicate the size of a given data download in advance of a data download workflow, when the size of the data is known or can be easily derived. |

| **Permission 2** | **/per/core/links** |
|---|---|
| A | A WNM links array property MAY provide link objects which reference APIs or Web Accessible Folders (WAF). |

```
"links": [{
  "href": "https://example.org/data/4Pubsub/92c557ef-d28e-4713-91af-
2e2e7be6f8ab.bufr4",
  "rel": "canonical",
  "type": "application/x-bufr"
}]
```

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

# Annex A: Conformance Class Abstract Test Suite (Normative)

## A.1. Conformance Class: Core

**label**

http://www.wmo.int/spec/wnm/1.0/conf/core

**subject**

Requirements Class "core"

**classification**

Target Type:Notification Metadata

### A.1.1. Validation

**label**

/conf/core/validation

**subject**

/req/core/validation

**test-purpose**

Validate that a WNM is valid to the authoritative WNM schema.

Run JSON Schema validation on the WNM against the WNM authoritative schema.

### A.1.2. Identifier

**label**

/conf/core/identifier

**subject**

/req/core/identifier

**test-purpose**

Validate that a WNM has a valid identifier.

Check for the existence of an `id` property in the WNM.

### A.1.3. Version

**label**

/conf/core/identifier

**subject**

/req/core/identifier

**test-purpose**

Validate that a WNM has a valid version.

Check for the existence of an `version` property in the WNM.

Check that the `id` property is equal to `v04`.

### A.1.4. Type

**label**

/conf/core/type

**subject**

/req/core/type

**test-purpose**

Validate that a WNM provides a valid type.

Check for the existence of a `type` property in the WNM

Check that the `type` property is equal to `Feature`.

### A.1.5. Geometry

**label**

/conf/core/geometry

**subject**

/req/core/geometry

**test-purpose**

Validate that a WNM provides a valid geometry property.

Check for the existence of one `geometry` property in the WNM.

Check that all `geometry.coordinates` value data types are integers or floats.

Check that `geometry.coordinates` longitudinal values are between -180 and 180.

Check that `geometry.coordinates` latitudinal values are between -90 and 90.

Check that `geometry` property is a valid GeoJSON geometry.

### A.1.6. Properties / Publication Time

**label**

/conf/core/pubtime

**subject**

/req/core/pubtime

**test-purpose**

Validate that a WNM has a valid pubtime.

Check for the existence of an `properties.pubtime` property.

Check that the `properties.pubtime` property is in RFC3339 format.

Check that the `properties.pubtime` property is in the UTC timezone.

### A.1.7. Properties / Data Identification

**label**

/conf/core/data_id

**subject**

/req/core/data_id

**test-purpose**

Validate that a WNM has a valid data_id.

Check for the existence of an `data_id` property.

Check that the `data_id` property starts with a valid WIS2 topic with the channel and version removed.

### A.1.8. Properties / Temporal description

**label**

/conf/core/temporal

**subject**

/req/core/temporal

**test-purpose**

Validate that a WNM provides a valid temporal description.

Check for the existence of one `properties.datetime` or both `properties.start_datetime` and `properties.end_datetime`.

Check that the any of the `properties.datetime` or both `properties.start_datetime` and `properties.end_datetime` properties are in RFC3339 format.

Check that the any of the `properties.datetime` or both `properties.start_datetime` and `properties.end_datetime` properties are in the UTC timezone.

### A.1.9. Links

**label**

/conf/core/links

**subject**

/req/core/links

**test-purpose**

Validate that a WNM provides a link array property.

Check for the existence of a single `links` array property in the WNM.

Check that the `links` array property provides a minimum of one link object.

For each link object, check that the `rel` property contains a valid IANA link relation [23].

Check that the `links` array property contains one link object with a `rel` property with value `canonical` as well as an `href` property.

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

# Annex B: Schemas (Normative)

| NOTE | The schema document will only be published on `schemas.wmo.int` once the standard has been approved. |
|---|---|

## B.1. WIS2 Notification Message Schema

```
$schema: https://json-schema.org/draft/2020-12/schema
$id: https://schemas.wmo.int/wis2/broker/message/0.9.0/schema.yml
title: WMO WIS 2.0 broker message schema
description: WMO WIS 2.0 broker message schema

definitions:
  schemas:
    link:
      # from https://github.com/opengeospatial/ogcapi-
features/blob/master/core/openapi/schemas/link.yaml
      type: object
      required:
        - href
        - rel
      properties:
        href:
          type: string
          example: https://example.com/data/obs/123
        rel:
          type: string
          description: |
            the link relation describing the relationship between the link and the
message.
            See https://www.iana.org/assignments/link-relations/link-relations.xhtml
for the
            official list of IANA link relations.
          example: canonical
        type:
          type: string
          example: application/geo+json
        hreflang:
          type: string
          example: en
        title:
          type: string
          example: Trierer Strasse 70, 53115 Bonn
        length:
          type: integer
          description: for file links optional length provides the number of bytes in
the file
    pointGeoJSON:
```

```
      # from https://raw.githubusercontent.com/opengeospatial/ogcapi-
features/master/core/openapi/schemas/pointGeoJSON.yaml
      type: object
      required:
        - type
        - coordinates
      properties:
        type:
          type: string
          enum:
            - Point
        coordinates:
          type: array
          minItems: 2
          items:
            type: number
    polygonGeoJSON:
      # from https://raw.githubusercontent.com/opengeospatial/ogcapi-
features/master/core/openapi/schemas/polygonGeoJSON.yaml
      type: object
      required:
        - type
        - coordinates
      properties:
        type:
          type: string
          enum:
            - Polygon
        coordinates:
          type: array
          items:
            type: array
            minItems: 4
            items:
              type: array
              minItems: 2
              items:
                type: number

type: object
properties:
  id:
    type: string
    format: uuid
    description: UUID (RFC4122) - Guarantee uniqueness of the message over (at least)
a 24h period
  version:
    type: string
    description: Version of message specification
    const: v04
  type:
```

```
    type: string
    enum:
      - Feature
  geometry:
    oneOf:
      - type: 'null'
      - $ref: '#/definitions/schemas/pointGeoJSON'
      - $ref: '#/definitions/schemas/polygonGeoJSON'
  properties:
    type: object
    properties:
      pubtime:
        type: string
        format: date-time
        description: |
          Identifies the date/time of when the file was posted/published, in RFC3339
format.
          The publication date/time is critical for subscribers to prevent message
loss by knowing
          their lag (how far behind the publisher they are).
      data_id:
        type: string
        description: |
          Uniquely identifies the data. It is formed of two parts:
          1. MQTT topic hierarchy where this message will be published as defined in
https://github.com/wmo-im/wis2-topic-hierarchy without leading channel/version
          2. a unique identifier of the data. It could be the filename used by the
originating center or a unique UUID (RFC4122) or anything chosen
          by the originating center as long as it is unique over a 1 week period.
      datetime:
        type: string
        format: date-time
        description: Identifies the date/time of the data being published, in RFC3339
format.
      start_datetime:
        type: string
        format: date-time
        description: Identifies the start date/time date of the data being published,
in RFC3339 format.
      end_datetime:
        type: string
        format: date-time
        description: Identifies the end date/time date of the data being published, in
RFC3339 format.
      integrity:
        type: object
        description: Specifies a checksum to be applied to the data to ensure that the
download is accurate.
        properties:
          method:
            type: string
```

```
            description: |
                A specific set of methods for calculating the checksum algorithms:
                * ``sha256``: the Secure Hash Algorithm 256 bits, value is base64
encoded.
                * ``sha512``: the Secure Hash Algorithm 512 bits, value is base64
encoded.
                * ``arbitrary``: an arbitrary string is used to identify the value.
                * ``md5``: the Message Digest 5 hash (obsolete, perhaps will be
rejected)
            enum:
              - sha256
              - sha512
              - arbitrary
              - md5
          value:
            type: string
            description: checksum value.
        required:
          - method
          - value
      content:
        type: object
        description: Used to embed small products inline within the message.
        properties:
          encoding:
            type: string
            description: encoding of content
            enum:
              - utf-8
              - base64
              - gzip
          size:
            type: integer
            description: Number of bytes contained in the file. Together with the
``integrity`` property, it provides additional assurance that file content was
accurately received.
          value:
            type: string
            description: the inline content of the file.
        required:
          - encoding
          - size
          - value
    required:
      - pubtime
      - data_id
    oneOf:
      - allOf:
        - required:
          - start_datetime
          - end_datetime
```

```
      - allOf:
        - required:
          - datetime

  links:
    type: array
    minItems: 1
    items:
      $ref: '#/definitions/schemas/link'
required:
  - id
  - version
  - type
  - geometry
  - properties
  - links
example:
  $ref: https://raw.githubusercontent.com/wmo-im/wis2-notification-
message/main/examples/example1.json
```

# Annex C: Examples (Informative)

## C.1. WIS2 Notification Message Examples

*Example: single observation including meaningful geometry, observation datetime and content*


```
{
    "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"
        }
    ]
}
```

*Example: NWP product including polygon* `geometry`, `start_datetime` *and* `end_datetime`. *The polygon is an array with 5 elements. The first (and last) elements are most southwestern point of the rectangle using longitude and latitude and followed by other coordinates in a clockwise direction.*


```
{
    "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"
        }
    ]
}
```

*Example: minimal viable message compliant to the specification*


```
{
    "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"
        }
    ]
}
```

# Annex D: Bibliography


- W3C/OGC: Spatial Data on the Web Best Practices, W3C Working Group Note 28 September 2017, https://www.w3.org/TR/sdw-bp
- W3C: Data on the Web Best Practices, W3C Recommendation 31 January 2017, https://www.w3.org/TR/dwbp
- IANA: Link Relation Types, https://www.iana.org/assignments/link-relations/link-relations.xml

# Annex E: Revision History

| Date | Release | Editor | Primary clauses modified | Description |
|---|---|---|---|---|
| 2022-12-05 | Template | Tom Kralidis | all | update from TT-Protocols/ET-W2AT |