WIS2 Cookbook

World Meteorological Organization

Date: 2024-11-13

Version: 2024-04-16

Document location: https://community.wmo.int/wis2-cookbook

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

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

Table of Contents

Chapter 1. Introduction

1.1. Purpose

In conjunction with the Manual on the WMO Information System volume II (WMO-No. 1060) (Manual on WIS volume II: WIS 2.0), the present Guide to the WMO Information System volume II (Guide to WIS volume II: WIS 2.0) is designed to ensure adequate uniformity and standardization in the data, information and communication practices, procedures and specifications employed by Members of the World Meteorological Organization (WMO) in the operation of the WMO Information System WIS 2.0 as it supports the mission of the Organization. The Manual on WIS contains standard and recommended practices, procedures and specifications. The Guide to WIS contains additional information concerning practices, procedures, and specifications that Members are invited to follow or implement in establishing and conducting their arrangements in compliance with the WMO Technical Regulations and in developing meteorological and hydrological services.

This cookbook provides various code snippets, recipes and workflow examples in support of WIS2 requirements. This cookbook is a working document; contributions are encouraged and can be added via GitHub.

[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

Chapter 2. Recipes for data consumers

2.1. Search the Global Discovery Catalogue

The Global Discovery Catalogue (GDC) allows for a wide range of query predicates to search for data in WIS2 as per the OGC API - Records - Part 1: Core specification.

The GDC can be searched via the `/collections/wis2-discovery-metadata/items` endpoint. This endpoint provides a number query parameters as described in the examples below.

NOTE: examples below are not URL encoded for clarity / readability, but should be when interacting with the GDC.

2.1.1. Spatial queries

- search for metadata records of data in Canada: `bbox=-142,42.-52,84`

2.1.2. Temporal queries

- search for metadata records updated since 29 July 2024: `datetime=2024-07-29/..`
- search for metadata records updated before 29 July 2024: `datetime=../2024-07-29`
- search for metadata records updated on 29 July 2024: `datetime=2024-07-29`

2.1.3. Equality queries

- search for metadata records whose title contains the terms hourly observations: `title=hourly observations`
- search for metadata records whose title contains the terms hourly or observations: `title=hourly | observations`

2.1.4. Freetext search

- search metadata records for temperature: `q=temperature`
- search metadata records for GRIB2 data: `q=GRIB2`
- search metadata records for any GRIB data: `q=*GRIB*`
- search for either GRIB data or data in Europe with subscriptions to the Météo-France Global Broker: `q=(*GRIB* OR *Europe*) AND *globalbroker*`
- search for data from Belize with MQTT subscription capabilitiesi: `q="cache/a/wis2/bz-nms"`

2.1.5. Sorting

- sort search results by title, ascending: `sortby=title`
- sort search results by title, descending: `sortby=-title`

2.1.6. Paging

- present search results 1-10: `limit=10`
- present search results 11-20: `limit=10&offset=10`
- limit to 3 search results: `limit=3`

Chapter 3. Recipes for data publishers

3.1. Validate a WIS2 Notification Message

A WIS2 Notification Message provides a JSON Schema which can be used by any programming language that supports JSON and JSON Schema validation.

Using Python and check-jsonschema

```
# install check-jsonschema Python Package from the Python Package Index (PyPI)
pip3 install check-jsonschema

# download WNM schema
curl -O http://schemas.wmo.int/wnm/1.0.0/schemas/wis2-notification-message-
bundled.json

# run schema validation
check-jsonschema --schemafile wis2-notification-message-bundled.json
/path/to/my/wnm.json
```

The pywis-pubsub tool provides a test suite to validate a message against the WNM specification requirements, as well as a Python API for application integration. Consult the pywis-pubsub README on GitHub for more information/examples.

Using pywis-pubsub

```
# install pywis-pubsub
pip3 install pywis-pubsub

# sync WIS2 notification schema
pywis-pubsub schema sync

# validate WNM against abstract test suite (file on disk)
pywis-pubsub ets validate /path/to/file.json

# validate WNM against abstract test suite (URL)
pywis-pubsub ets validate https://example.org/path/to/file.json
```

3.2. Publish a WIS2 Notification Message with access control

Recommended data in WIS2 may be open or access controlled. For data publication with access control implications, WNM provides a `security` object as part a link object. The `security` object is defined using OpenAPI Security Scheme definitions.

Access control using HTTP Basic authentication

```
{
  "rel": "canonical",
  "type": "application/grib2",
  "href": "https://example.org/my/protected/data/nwp/12/003/20240805120000-air-temp-
500.grib2",
  "security": {
    "default": {
      "type": "http",
      "scheme": "basic",
      "description": "Please contact us for access information"
    }
  }
}
```

Access control using an API key

```
{
  "rel": "canonical",
  "type": "application/geo+json",
  "href": "https://example.org/my/protected/data/nwp/12/003/20240805120000-air-temp-
500.grib2",
  "security": {
    "default": {
      "type": "apiKey",
      "name": "api-key",
      "in": "query",
      "description": "Please see https://example.org/contact-us for more information"
    }
  }
}
```

Note:

- the child propery under `security` (`default` in the examples above) can be any text or label. We use `default` here as a convention
- the `security.default.name` is the name of the API key parameter as defined by your API service
- only properties defined in the OpenAPI Security Scheme defintion are allowed. Any additional properties will invalidate the WNM

Of course, always ensure your WNM is valid (see Validate a WIS2 Notification Message for more information).

3.3. Validate a WMO Core Metadata Profile record

The [pywcmp](https://github.com/wmo-im/pywcmp) tool provides a test suite to validate a message against the WCMP2 specification requirements, as well as a Python API for application integration.

Consult the pywcmp README on GitHub for more information/examples.

Using pywcmp

```
# install pywcmp
pip3 install pywcmp

# sync WCMP2 schemas and codelists
pywis-pubsub bundle sync

# validate WCMP2 against abstract test suite (file on disk)
pywcmp ets validate /path/to/file.json

# validate WCMP2 against abstract test suite (URL)
pywcmp ets validate https://example.org/path/to/file.json
```

A WCMP2 record can also be validated using pywcmp "as a service" using the Canadian WIS2 Global Discovery Catalogue, which provides an online validator:

- Navigate to https://wis2-gdc.weather.gc.ca/openapi?f=html
- Navigate to section **pywcmp-wis2-wcmp2-ets**, endpoint `/processes/pywcmp-wis2-wcmp2-ets/execution` (POST)
- Click "Try it out"
- In the section "Mandatory execute request JSON", paste the WCMP2 JSON inside the `record` object

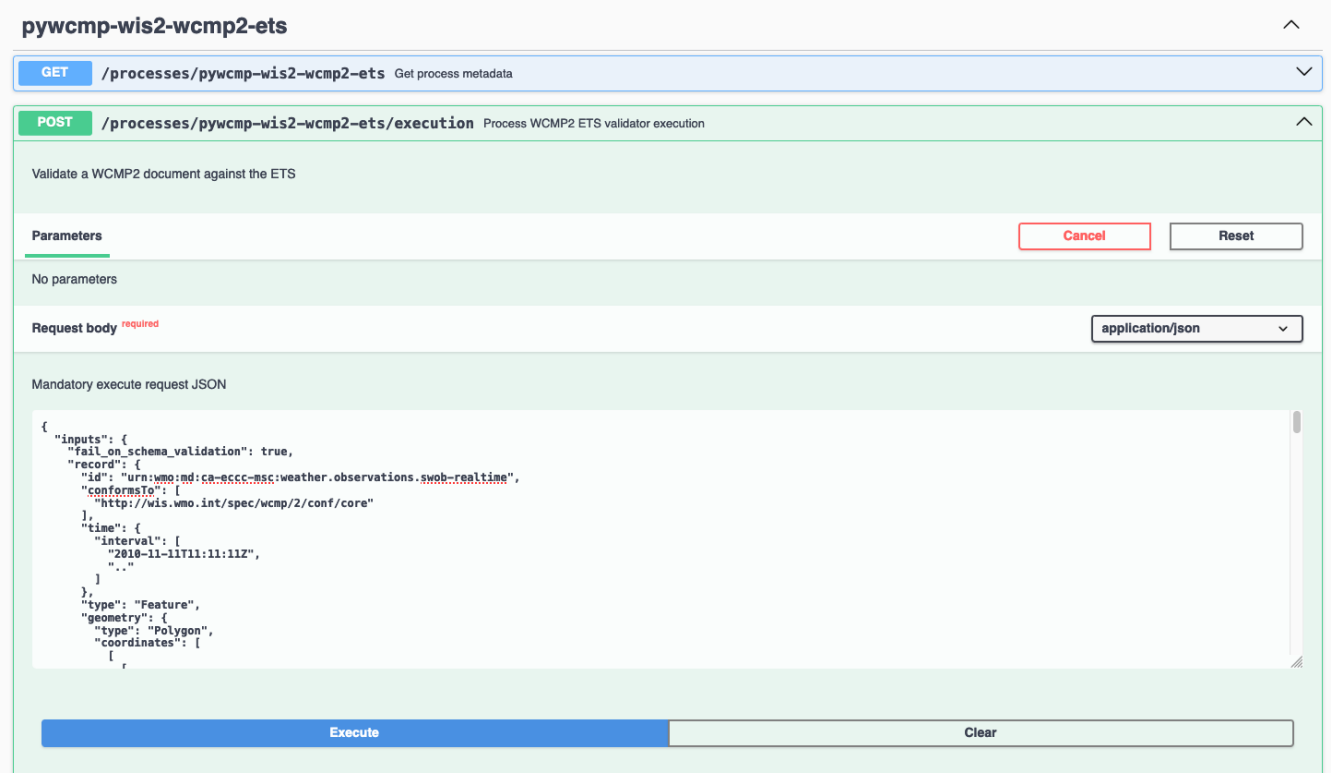

Figure 1. WIS2 GDC online validator, request

- Click "Execute"

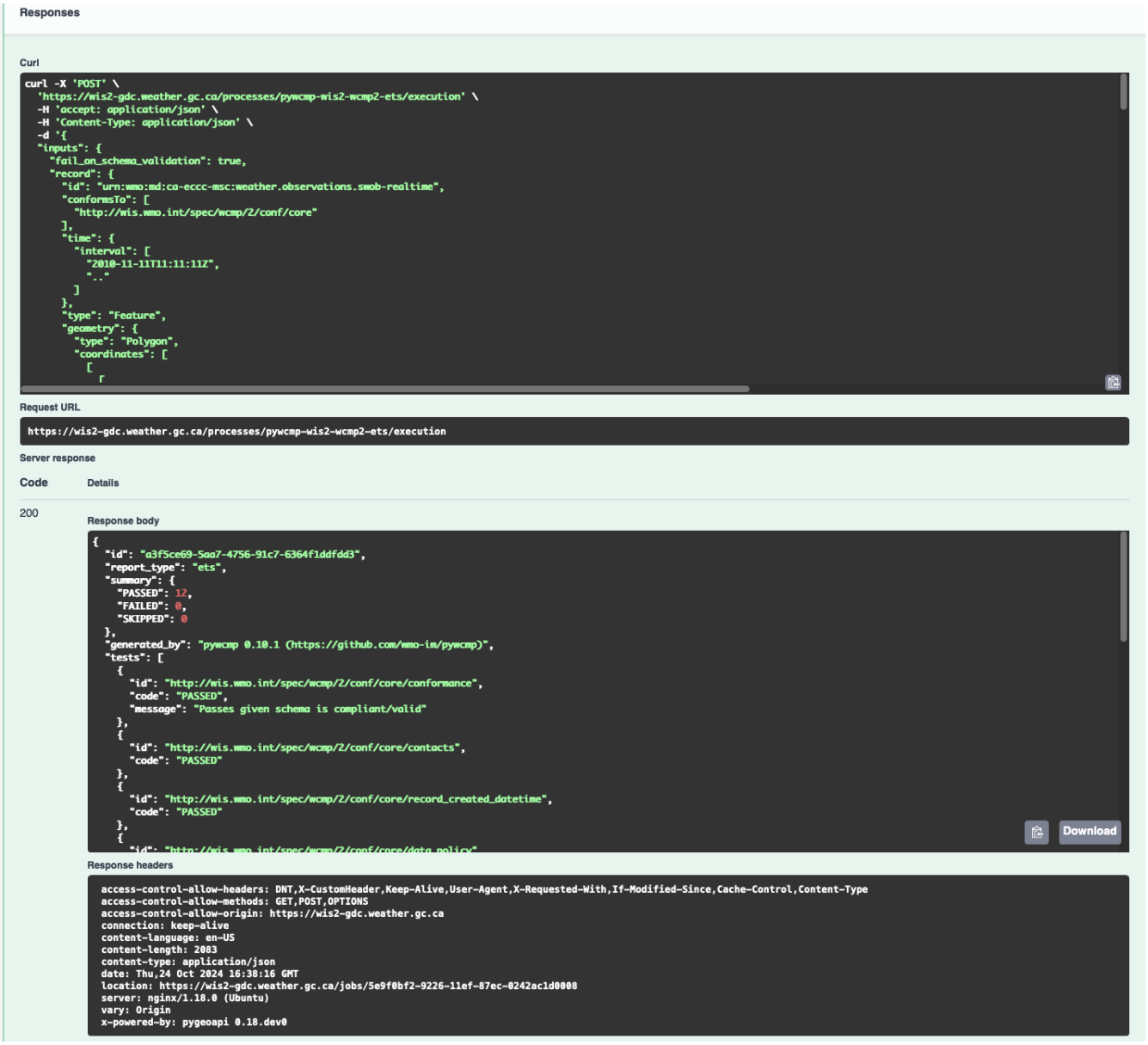

Figure 2. WIS2 GDC online validator, response

A response will be provided with validation results.

3.4. Advertise client side filters for data subscriptions in WCMP2 and WNM

A key concept of a WCMP2 record is "actionable links"; this means being able to access a dataset or data granule without any further interactions. For real-time data, a WCMP2 record provides linkages to the WIS2 Global Broker via the MQTT protocol. At its core, MQTT has two key components:

- topic: the topic to subscribe to
- message payload: the message provided as part of a notification to a given topic

WIS2 defines the WIS2 Topic Hierarchy (WTH) and WIS2 Notification Message (WNM) standards which provide a standards-based GeoJSON payload/message.

A typical MQTT link in a WCMP2 document is defined as follows:

Typical WCMP2 MQTT link

```
{
  "rel"  : "items",
  "type" : "application/geo+json",
  "title": "WIS2 notification service",
  "href" : "mqtts://example.org",
  "channel": "cache/a/wis2/ca-eccc-msc/data/core/weather/surface-based-
observations/synop"
}
```

Given WCMP2, WTH and WNM, a user can subscribe to topics related to data of interest for download and access.

In some cases, a dataset may be organized in a manner which requires additional further "filtering" such that a data consumer is only interested in a certain subset of the data granules being advertised by a given WNM. Some examples include (but are not limited to), where a data consumer may be only be interested in:

- surface weather observations from a certain station, or
- numerical weather prediction forecast data for a certain timestep or weather parameter

To implement this behaviour, add additional properties to both WCMP2 and WNM as follows:

3.4.1. Example: Surface weather observations

Surface weather observations: WCMP2 MQTT link with additional properties

```
{
  "rel"  : "items",
  "type" : "application/geo+json",
  "title": "Real-time notifications",
  "href" : "mqtts://globalbroker.meteo.fr:8883",
  "channel": "cache/a/wis2/ca-eccc-msc/data/core/weather/surface-based-
observations/synop",
  "properties": {
    "wigos_station_identifier": {
       "type": "string",
       "title": "WIGOS station identifier"
    }
  }
}
```

Surface weather observations: WNM additional properties

```
{
  "properties": {
```

```
    "wigos_station_identifier": "0-20000-0-71628"
}
```

When implemented by a data producer, a data consumer can:

- subscribe to real-time notifications to the given topic
- perform client side filtering by against all incoming WNMs with `properties.wigos_station_identifier = "0-20000-0-71628"`

3.4.2. Example: Numerical weather prediction based forecast

Numerical weather prediction: WCMP2 MQTT link with additional properties

```
{
  "rel"  : "items",
  "type" : "application/geo+json",
  "title": "Real-time notifications",
  "href" : "mqtts://globalbroker.meteo.fr:8883",
  "channel": "origin/a/wis2/ca-eccc-msc/data/core/weather/prediction/forecast/medium-
range/deterministic/global",
  "properties": {
    "model_run": {
       "type": "string",
       "title": "Model run",
       "enum": [
           "00",
           "12"
        ],
        "example": "00"
    },
    "forecast_hour": {
       "type": "string",
       "title": "Forecast hour",
        "example": "004"
    }
  }
}
```

Numerical weather prediction: WNM additional properties

```
{
  "properties": {
    "model_run": "00",
    "forecast_hour": "004"
}
```

A data producer would extend WCMP2 and WNM as follows:

- WCMP2: add a link `properties` object for MQTT links, where each key of the link `properties`

object is a JSON Schema property definition

- WNM: add additional properties (key: value pairs) in the `properties` object as desired

When implemented by a data producer, a data consumer can:

- subscribe to real-time notifications to the given topic
- perform client side filtering against all incoming WNMs with `properties.model_run = "00" and properties.forecast_hour = "004"`

A sample Python script can be found below. The script connects to the Météo-France Global Broker, subscribed to weather notifications from Environment and Climate Change Canada, Meteorological Service of Canada. The script then performs client side filtering by evaluating (for each WNM) the `properties.wigos_station_identifier` value to match a particular station (`0-20000-0-71628`).

Sample Python script to perform client side filtering

```
import json
from paho.mqtt import client as mqtt_client

broker = 'globalbroker.meteo.fr'
port = 8883
username = 'everyone'
password = 'everyone'
topic = 'cache/a/wis2/ca-eccc-msc/data/core/weather/surface-based-observations/synop'

wsi_to_filter = '0-20000-0-71628'


def connect_mqtt() -> mqtt_client:
    def on_connect(client, userdata, flags, reason_code, properties):
        if reason_code == 0:
            print(f'Connected to {broker}')
        else:
            print(f'Failed to connect: {reason_code}')

    def on_log(client, userdata, level, message):
        print("LOG:", message)

    client = mqtt_client.Client(mqtt_client.CallbackAPIVersion.VERSION2,
                                client_id='s123')
    client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.on_log = on_log
    client.tls_set(tls_version=2)
    client.connect(broker, port)

    return client


def subscribe(client: mqtt_client):
```

```
    def on_message(client, userdata, message):
        message_dict = json.loads(message.payload.decode())

        print('Performing client side filtering')
        wsi = message_dict['properties'].get('wigos_station_identifier')

        if wsi != wsi_to_filter:
            print(f'Topic: {message.topic}')
            print(f'Payload: {message.payload.decode()}')

    client.subscribe(topic)
    client.on_message = on_message


def run():
    client = connect_mqtt()
    subscribe(client)
    client.loop_forever()


if __name__ == '__main__':
    run()
```

Chapter 4. Recipes for Global Service operators