

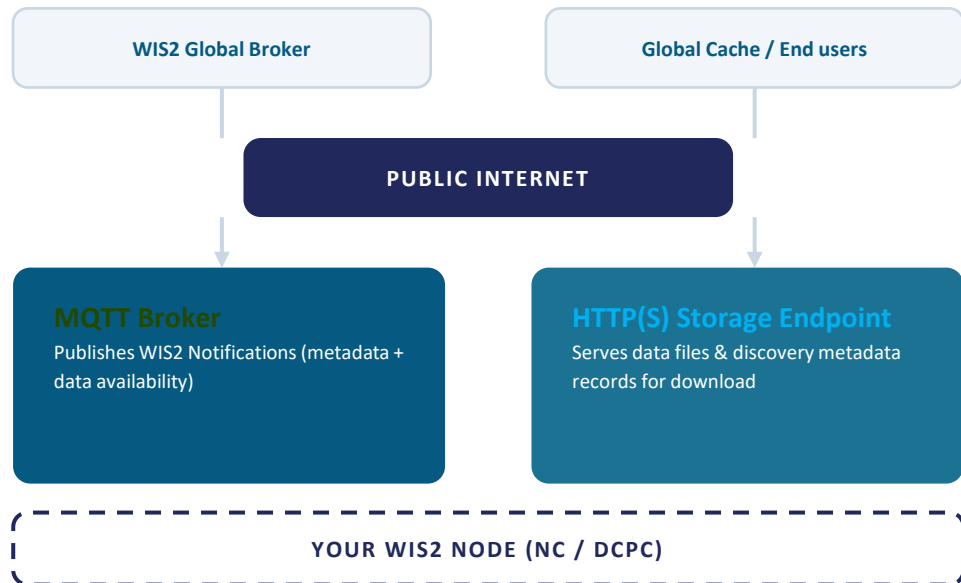


Implementation of own WIS2 Node - common issues, assessment-tools -

What is a WIS2 Node ?

A WIS2 node exposes 2 endpoints over the public internet:

- **MQTT broker**: to publish WIS2-notifications for metadata and data
- **HTTP storage endpoint**: to enable the download of data-files and metadata records



MQTT Broker requirements

- MQTT 5.0, secured as MQTTS on port 8883
- Password-controlled; IP-filtered for Global Brokers
- Publishes at QoS 1; topics not configured to retain
- Notifications = valid GeoJSON WIS2 Notification Messages (WNM)

HTTP storage requirements

- Reachable over both HTTP and HTTPS
- Serves core data (pulled by Global Caches) and recommended data (access-controlled)
- Serves WCMP2 discovery metadata records
- “canonical” links in each WNM point here

Reference implementation: wis2box ships both — Mosquitto for the broker, MinIO for storage — as a single-node Docker Compose stack.

MQTT Broker Options

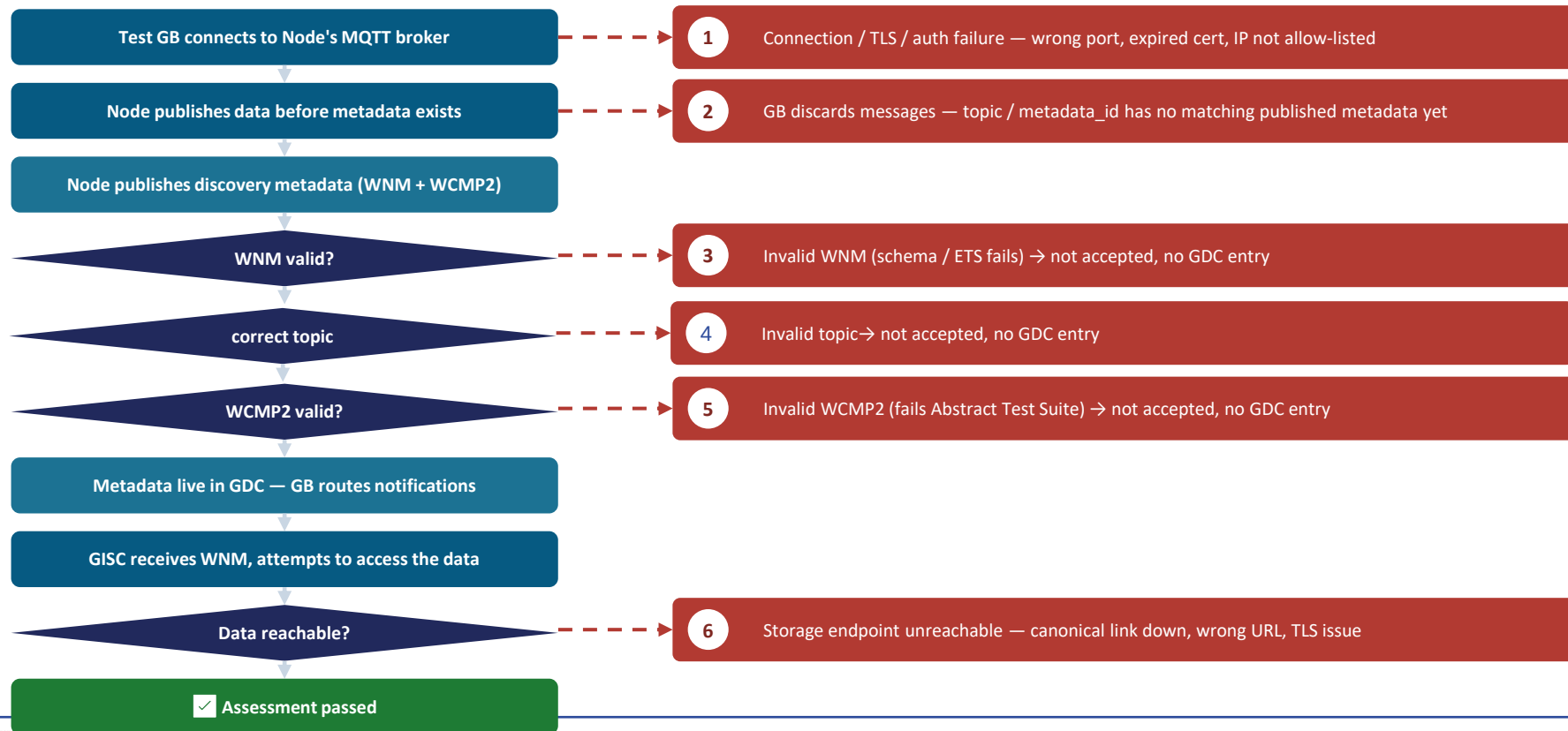
Any MQTT 5.0 broker that supports MQTTS, auth/ACLs and QoS 1 can serve as the Node's notification endpoint.

Broker	Language	License	Open source?	Clustering?
Eclipse Mosquitto	C	EPL / EDL	Open source	No (bridging only)
EMQX	Erlang	BSL 1.1 (since v5.9, May 2025)	Not OSI open source	Yes (paid; single-node free)
VerneMQ	Erlang	Apache 2.0	Open source	Yes (built-in, masterless)
HiveMQ CE	Java	Apache 2.0	Open source	No (Enterprise only)
RabbitMQ + MQTT plugin	Erlang	MPL 2.0	Open source	Yes (built-in)
NanoMQ	C	Apache 2.0	Open source	Partial (edge bridging)

Client-side integration libraries: **Python** — paho-mqtt, pywis-pubsub, Flask/FastAPI • **Node.js** — mqtt.js • **Go** — paho.mqtt.golang • **Java** — Eclipse Paho.



Node assessment - Stages and Error classes



1

Error Class 1 — Connection / TLS / Auth Failure

Occurs at: Step 1 — test GB connects to the Node's MQTT broker

WHAT HAPPENS

The test Global Broker cannot open or hold an MQTT session with the Node's broker, so nothing downstream ever happens — no notifications are ever seen on the network.

COMMON CAUSES

- Broker only exposed on 1883 (plain MQTT) or on the Docker-internal network — 8883/MQTTS never reaches the public interface
- TLS certificate expired, self-signed without being trusted, or missing intermediate certs in the chain
- Firewall / security group doesn't allow the GISC's or Global Broker's IP range on port 8883
- Auth configured but the GB's credentials were never created, or ACL doesn't grant read on origin/a/wis2/#

✓ HOW TO CHECK IT YOURSELF

Test the port is reachable from outside your

```
nc -zv <your-host> 8883
```

Inspect the TLS handshake and certificate chain:

```
openssl s_client -connect  
<host>:8883 -servername <host>
```

Subscribe as an external, unauthenticated-adjacent client:

```
mosquitto_sub -h <host> -p 8883 --cafile  
ca.crt -u everyone -P everyone -t '#' -v
```

Check broker logs or for rejected connections / bad cert errors.



2

Error Class 2 — Data Published Without Metadata

Occurs at: Step 2 — Node publishes a data notification before/without matching discovery metadata

WHAT HAPPENS

The Global Broker (and any subscriber) receives a perfectly well-formed WNM for a data file, but has no WCMP2 discovery metadata for that dataset — so the GB discards it

COMMON CAUSES

- Data ingestion pipeline went live before the WCMP2 discovery metadata record was published to the GDC
- Dataset removed or renamed in your Node's catalogue without updating data_id / metadata_id in the ingest configuration

✓ HOW TO CHECK IT YOURSELF

Pull a live sample WNM and read its metadata_id:

```
pywis-pubsub subscribe --config  
local.yml
```

Query the for that exact id:

```
curl  
"https://gdc.wis2dev.io/collections/wis2  
-discovery-metadata/items?q=<centre-id>"
```

In wis2box: confirm the dataset shows up in the wis2box-webapp Dataset tab before you start ingesting.

3

Error Class 3 — Invalid WNM (schema / ETS)

Occurs at: Metadata publish step — WIS2 Notification Message validation

WHAT HAPPENS

The notification message itself fails JSON Schema validation or the WNM Abstract Test Suite, so it is rejected before it can ever be linked to discovery metadata or routed further.

COMMON CAUSES

- Missing required properties — pubtime, datetime, or data_id left out
- geometry coordinates in the wrong order (must be [longitude, latitude, elevation?], not lat/lon)
- datetime / pubtime not valid ISO 8601, or conformsTo missing/incorrect
- security object on a link uses fields outside the OpenAPI Security Scheme keywords — the whole message is invalidated

✓ HOW TO CHECK IT YOURSELF

Sync the schema once, then validate any message

```
pywis-pubsub schema sync && pywis-pubsub  
message validate msg.json
```

Run the full Abstract Test Suite (catches logic rules, not just schema):

```
pywis-pubsub ets validate msg.json
```

Or validate with a generic JSON Schema tool:

```
check-jsonschema --schemafile wis2-  
notification-message-bundled.json  
msg.json
```

Eyeball raw payloads live in MQTT Explorer before they even reach validation.



3

Error Class 3 — Invalid WNM (schema / ETS)

Occurs at: Metadata publish step — WIS2 Notification Message validation

WRONG

Missing required field pubtime.

```
...  
"properties": {  
  "datetime": "2026-08-05T12:00:00Z",  
  "data_id": "exercise/synop/ex02.bufr4",  
  "metadata_id": "urn:wmo:md:xx-training-  
centre:observations.synop"  
},  
...
```

✓ CORRECTED

Add pubtime. Run pywis-pubsub ets validate before publishing to catch this automatically, every time.

```
...  
"properties": {  
  "datetime": "2026-08-05T12:00:00Z",  
  "pubtime": "2026-08-05T12:05:00Z",  
  "data_id": "exercise/synop/ex02.bufr4",  
  "metadata_id": "urn:wmo:md:xx-training-  
centre:observations.synop"  
},  
...
```

3

Error Class 3 — Invalid WNM (schema / ETS)

Occurs at: Metadata publish step — WIS2 Notification Message validation

WRONG

Missing required field datetime.

```
...
"properties": {
  "pubtime": "2026-08-05T12:05:00Z",
  "data_id": "exercise/synop/ex02.bufr4",
  "metadata_id": "urn:wmo:md:xx-training-
centre:observations.synop"
},
...
```

✓ CORRECTED

Add datetime, it could be null, but the field is required. Run pywis-pubsub ets validate before publishing to catch this automatically, every time.

```
...
"properties": {
  "pubtime": "2026-08-05T12:05:00Z",
  "datetime": null,
  "data_id": "exercise/synop/ex02.bufr4",
  "metadata_id": "urn:wmo:md:xx-training-
centre:observations.synop"
},
...
```

3

Error Class 3 — Invalid WNM (schema / ETS)

Occurs at: Metadata publish step — WIS2 Notification Message validation

WRONG

Invalid geometry format.

```
...  
  "geometry": {  
    "type": "Point",  
    "coordinates": {  
      "lat": 60.1268,  
      "lon": 11.3933  
    }  
  }  
...
```

✓ CORRECTED

The geometry object needs to be a valid GeoJSON format. Run pywis-pubsub ets validate before publishing to catch this automatically, every time.

```
...  
  "geometry": {  
    "type": "Point",  
    "coordinates": [  
      60.1268,  
      11.3933  
    ]  
  },  
...
```

3

Error Class 3 — Invalid WNM (schema / ETS)

Occurs at: Metadata publish step — WIS2 Notification Message validation

WRONG

Sending WCMP2 as WNM

```
...  
  "conformsTo": [  
    "http://wis.wmo.int/spec/wcmp/2/conf/core"  
  ],  
...
```

✓ CORRECTED

Correct conformsTo. Run pywis-pubsub ets validate before publishing to catch this automatically, every time.

```
...  
  "conformsTo": [  
    "http://wis.wmo.int/spec/wnm/1/conf/core"  
  ],  
...
```

4

Error Class 4 — Invalid Topic

Occurs at: Checked alongside data/metadata publication — MQTT topic vs. WIS2 Topic Hierarchy

WHAT HAPPENS

The MQTT topic a message is published on doesn't conform to the WIS2 Topic Hierarchy (WTH), so subscribers filtering by topic never see it — even though the payload itself may be perfectly valid.

COMMON CAUSES

- centre-id not lower-case (topics are case-sensitive; WIS2 centre-ids must be lower-case)
- Extra, undefined sub-levels added beyond what's defined for the Earth system discipline (only experimental allows free-form sub-topics)
- Wrong top-level prefix — publishing under cache/a/wis2/... instead of origin/a/wis2/... (a Node always publishes as origin)
- Topic actually published on doesn't match the channel advertised in the WCMP2 record's items link — subscribers using the advertised topic get nothing

✓ HOW TO CHECK IT YOURSELF

Validate a topic against the published hierarchy:

```
pywis-topics bundle sync && pywis-topics  
topic validate <topic>
```

Watch the live topic string in MQTT Explorer and compare byte-for-byte with the WCMP2 channel value.

Cross-check discipline/sub-level spelling against the WMO Codes Registry:

```
https://codes.wmo.int/wis/topic-  
hierarchy
```

Keep to ≤ 4 additional sub-levels — anything deeper is usually a sign the topic was over-engineered.



4

Error Class 4 — Invalid Topic

Occurs at: Checked alongside data/metadata publication — MQTT topic vs. WIS2 Topic Hierarchy

X WRONG

Centre-id published in mixed case, plus an undefined station-level sub-topic tacked on beyond what's defined for surface-based-observations.

```
origin/a/wis2/XX-Centre/data/core/weather/  
surface-based-observations/synop/station123
```

✓ CORRECTED

Lower-case the centre-id and stop at the defined topic level.
Need station-level filtering? Put wigos_station_identifier in the WNM's properties instead

```
origin/a/wis2/xx-centre/data/core/weather/  
surface-based-observations/synop
```

4

Error Class 4 — Invalid Topic

Occurs at: Checked alongside data/metadata publication — MQTT topic vs. WIS2 Topic Hierarchy

X WRONG

undefined sub-topic for surface-based-observations.

```
origin/a/wis2/xx-centre/data/core/weather/  
surface-based-observations/dropsonde
```

✓ CORRECTED

Also dropsonde goes to temp topic. Only topics defined in <https://wmo-im.github.io/wis2-topic-hierarchy/standard/wis2-topic-hierarchy-STABLE.html> are valid.

```
origin/a/wis2/xx-centre/data/core/weather/  
surface-based-observations/temp
```

5

Error Class 5 — Invalid WCMP2

Occurs at: Metadata publish step — WCMP2 discovery metadata validation

WHAT HAPPENS

The WNM itself is valid, but the WCMP2 discovery metadata record it points to fails the WCMP2 Abstract Test Suite, so the GDC never ingests it

COMMON CAUSES

- Missing mandatory elements — time, links, or contacts incomplete or absent
- id / metadata identifier doesn't follow the urn:wmo:md:<centre-id>:<local-id> pattern expected by the GDC or use a invalid centre-id

✓ HOW TO CHECK IT YOURSELF

Sync schemas/codelists once, then validate:

```
pywcmp bundle sync && pywcmp ets  
validate wcmp2.json
```

Run the (non-blocking) quality/KPI checks too:

```
pywcmp kpi validate wcmp2.json
```

**Or paste the record into the GDC's online validator
([wis2-gdc.weather.gc.ca](https://wis2-gdc.weather.gc.ca/wis2.dwd.de/openapi) / [wis2.dwd.de openapi](https://wis2.dwd.de/openapi) UI).**

**In wis2box: check the record actually appears in your
own discovery-metadata collection before assuming
the GDC is at fault.**

```
curl <your-  
host>/oapi/collections/discovery-  
metadata/items
```

5

Error Class 5 — Invalid WCMP2

Occurs at: Metadata publish step — WCMP2 discovery metadata validation

X WRONG

Missing channel

```
"links": [  
  {  
    "rel": "items",  
    "href":  
"mqtt://broker.example.org:8883",  
    "type": "application/json"  
  }  
]
```

✓ CORRECTED

Add valid topic hierarchy to channel field.

```
"links": [  
  {  
    "rel": "items",  
    "href":  
"mqtt://broker.example.org:8883",  
    "type": "application/json",  
    "channel": "origin/a/wis2/de-  
dwd/data/core/weather/surface-based-  
observations/synop"  
  }  
]
```

6

Error Class 6 — Storage Endpoint Unreachable

Occurs at: Step 8 — GISC / end user attempts to access the data via the canonical link

WHAT HAPPENS

Everything upstream worked — metadata is in the GDC, notifications are routing — but the actual file behind the WNM's canonical link can't be downloaded, so caches and end users get a failure at the last step.

COMMON CAUSES

- HTTP(S) storage endpoint not exposed publicly, or only reachable from inside the Node's own network
- href in the WNM still points at an internal hostname (e.g. minio:9000) instead of the public URL
- TLS certificate problem on the storage/reverse-proxy side, separate from the broker's certificate
- File already expired/cleaned up from storage before Global Caches had a chance to pull it, or access-control unexpectedly blocking anonymous access to core data

✓ HOW TO CHECK IT YOURSELF

Check the canonical link directly, from an external network:

```
curl -I <canonical-href>
```

Validate the message and try to actually fetch the referenced data:

```
pywis-pubsub message verify msg.json
```

Confirm the URL isn't a leftover internal/Docker hostname

In wis2box: check web-proxy logs / Grafana for 4xx/5xx spikes on /data/ requests.

Tool	Purpose	Typical command
GISC WIS2 Node assessment	Formal compliance check by the principal GISC before a Node goes operational	Manual process via GISC + WMO Secretariat
pywcmp	Validate & quality-assess WCMP2 discovery metadata (Abstract Test Suite + draft KPIs)	<code>pywcmp ets validate file.json</code> <code>pywcmp kpi validate file.json</code>
pywis-pubsub	Validate a WNM against schema + Abstract Test Suite; also subscribe/download	<code>pywis-pubsub ets validate file.json</code>
check-jsonschema	Generic JSON Schema validation against the published WNM schema	<code>check-jsonschema --schemafilename file.json</code> <code>wis2-notification-message-bundled.json file.json</code>
GDC online validator (CA / DE)	Web UI wrapping pywcmp ETS/KPI — paste JSON or a URL, no install needed	<code>wis2-gdc.weather.gc.ca/openapi?f=html</code>
WIS2 Monitoring Events	Pilot GDC validation/KPI reports pushed back to your centre via the Global Broker	Subscribe: <code>monitor/a/wis2/<centre-id></code>

1 · SETUP

Install

```
pip3 install pywis-pubsub
```

Fetch the current WNM JSON Schema

```
pywis-pubsub schema sync
```

2 · SUBSCRIBE & DOWNLOAD

Connect & echo live messages

```
pywis-pubsub subscribe --config  
local.yml
```

Also download the referenced data

```
pywis-pubsub subscribe --config  
local.yml --download
```

Filter messages by bounding-box geometry

```
pywis-pubsub subscribe --config  
local.yml --bbox=-142,42,-52,84
```

3 · VALIDATE & ASSESS

Schema validation of a saved message

```
pywis-pubsub message validate  
msg.json
```

Verify integrity of the downloaded data

```
pywis-pubsub message verify  
msg.json
```

Full Abstract Test Suite

```
pywis-pubsub ets validate  
file.json
```

KPI / quality assessment

```
pywis-pubsub kpi validate  
file.json
```

1 · SETUP

Install

```
pip3 install pywcmp
```

Sync WCMP2 schemas, codelists & topics

```
pywcmp bundle sync
```

2 · VALIDATE (ETS)

Validate a local WCMP2 record against the Abstract Test Suite

```
pywcmp ets validate file.json
```

Validate a record straight from its URL

```
pywcmp ets validate  
https://example.org/record.json
```

Run ETS logic checks even if schema validation would otherwise block them

```
pywcmp ets validate <url> --no-  
fail-on-schema-validation
```

3 · QUALITY (KPI)

Score record quality against the draft WCMP2 KPIs

```
pywcmp kpi validate file.json
```

See the per-criterion detail behind the score

```
pywcmp kpi validate file.json --  
verbosity DEBUG
```

Write the full report to a logfile for later review

```
pywcmp kpi validate <url> --  
verbosity DEBUG --logfile  
/tmp/foo.txt
```

check-jsonschema in practice

1 · SETUP

Install

```
pip3 install check-jsonschema
```

Sync WNM schemas

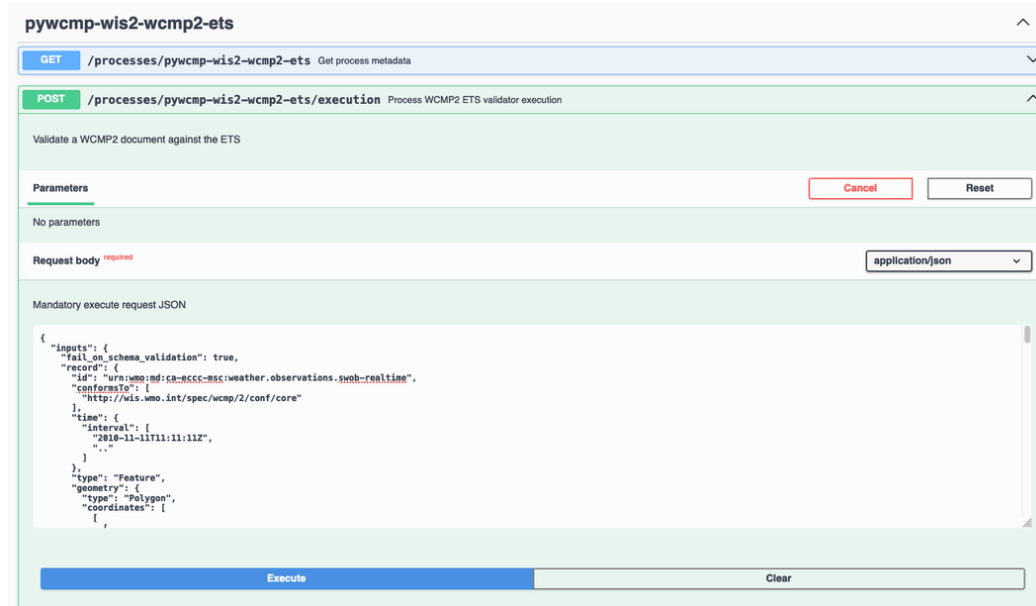
```
curl -O  
http://schemas.wmo.int/wnm/1.1.0/  
schemas/wis2-notification-  
message-bundled.json
```

2 · VALIDATE (ETS)

Validate a local WNM record against the schema

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

1. Navigate to <https://wis2-gdc.weather.gc.ca/openapi?f=html> or <https://wis2.dwd.de/gdc/openapi?f=html>
2. Navigate to section pywcmp-wis2-wcmp2-ets, endpoint `/processes/pywcmp-wis2-wcmp2-ets/execution` (POST)
3. Click "Try it out"
4. In the section "Mandatory execute request JSON", paste the WCMP2 JSON inside the `record` object
5. A response will be provided with validation results.



The screenshot shows the 'Try it out' interface for the `pywcmp-wis2-wcmp2-ets` endpoint. The selected method is `POST /processes/pywcmp-wis2-wcmp2-ets/execution` with the description 'Process WCMP2 ETS validator execution'. The 'Parameters' section is empty. The 'Request body' section is set to `application/json` and contains a JSON object for the 'record' field. The JSON object includes a 'fail_on_schema_validation' flag, a 'record' object with 'id', 'uri', and 'conformsTo' fields, and a 'time' object with 'interval' and 'coordinates' fields. The 'Execute' button is at the bottom.

```
{
  "inputs": {
    "fail_on_schema_validation": true,
    "record": {
      "id": "urn:wwm:md:ca-eccc-ssc:weather.observations.sweb-realtime",
      "conformsTo": [
        "http://wis.wwm.int/spec/wcmp/2/conf/core"
      ],
      "time": {
        "interval": [
          "2018-11-11T11:11:11Z",
          "..."
        ],
        "type": "Feature",
        "geometry": {
          "type": "Polygon",
          "coordinates": [
            [
              ...
            ]
          ]
        }
      }
    }
  }
}
```

WMO WIS2 Monitoring Events

WNM

➔ GB sends out warnings/erros about received WNM

```
{
  "id": "08361ecb-e7ff-4965-9abe-465b63433ca5",
  "specversion": "1.0",
  "type": "int.wmo.wis.wme.event.wnm.validation",
  "source": "io-wis2dev-global-broker",
  "subject": "io-wis2dev-12-test",
  "time": "2024-12-11T12:54:40.605Z",
  "datacontenttype": "application/json",
  "dataschema": "...",
  "data": {
    "conformsTo": [
      "http://wis.wmo.int/spec/wme/1/conf/monitoring-event-message-core"
    ],
    "severity": "ERROR",
    "content": {
      "channel": "cache/a/wis2/de-dwd-gts-to-wis2/data/recommended/U/A/N/T/99/KDDL",
      ...
    },
    "exception": {
      "code": "invalid-schema",
      "description": "WIS2 Notification Message not compliant with the defined schema",
      "errors": ...
    }
  }
}
```

WCMP2

➔ GDCs send out validation / KPI reports about published metadata records

```
{
  "id": "08361ecb-e7ff-4965-9abe-465b63433ca5",
  "specversion": "1.0",
  "type": "int.wmo.wis.wme.event.wnm.validation",
  "source": "io-wis2dev-global-broker",
  "subject": "io-wis2dev-12-test",
  "time": "2024-12-11T12:54:40.605Z",
  "datacontenttype": "application/json",
  "dataschema": "...",
  "data": {
    "conformsTo": [
      "http://wis.wmo.int/spec/wme/1/conf/monitoring-event-message-core"
    ],
    "severity": "INFO",
    "content": {
      "id": "f84f34d6-cfb0-4cff-98ec-32f88d0fd7b8",
      "title": "WCMP2 ETS report",
      "report_type": "ets",
      "summary": {
        "PASSED": 12,
        "FAILED": 0,
        "SKIPPED": 0
      }, ...
    }
  }
}
```

Thank you for your attention!