WIS2 Global Services testing

World Meteorological Organization

Date: 2025-05-06

Version: 2024-03-28

Document status: DRAFT

Document location: https://wmo-im.github.io/wis2-global-services-testing/global-services-testing/wis2-global-services-testing-DRAFT.html

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

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

Table of Contents

Chapter 1. Abstract

The subject of this Report is the results of testing and experimentation of WIS2 Global Services during the pre-operational phase of WIS2. Global Services testing is coordinated by the WIS2 Architecture and Transition team and provides results and recommendations on testing performance, availability and functionality.

[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. Executive Summary

TODO

Chapter 3. Scope

This report presents the testing framework put forth as part of the pre-operational phase of Global Services testing. This report also discussses the results and presents a set of conclusions and recommendations.

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.

4.1. Abbreviated terms

API

Application Programming Interface

GB

Global Broker

GC

Global Cache

GDC

Global Discovery Catalogue

GISC

Global Information System Centre GM: Global Monitor

HTTP

Hypertext Transfer Protocol

HTTPS

Hypertext Transfer Protocol Secure

JSON

JavaScript Object Notation

OGC

Open Geospatial Consortium

MQTT

Message Queuing Telemetry Transport

WCMP2

WMO Core Metadata Profile 2

WIS

WMO Information System

WMO

World Meteorological Organization

WNM

WIS2 Notification Message

WTH

WIS2 Topic Hierarchy

Chapter 5. References

- WMO: WMO Core Metadata Profile (2024) [1]
- WMO: WIS2 Notification Message (2024) [2]
- WMO: WIS2 Topic Hierarchy (2024) [3]
- Draft guidance on technical specifications of WIS2 (2024) [4]
- Draft guidance on transition from GTS to WIS2 (2024) [5]

[1] https://wmo-im.github.io/wcmp2

[2] https://wmo-im.github.io/wis2-notification-message

[3] https://wmo-im.github.io/wis2-topic-hierarchy

[4] https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html

[5] https://wmo-im.github.io/wis2-transition-guide/transition-guide/wis2-transition-guide-DRAFT.html

Chapter 6. High Level Architecture

The focus of testing is to evaluate functionality to ensure all WIS2 components perform as defined by the architecture. Testing is designed to enable core workflows:

- WIS2 Nodes providing data and metadata
- WIS2 Global Brokers subscribing to WIS2 Nodes
- WIS2 Global Caches providing data and metadata for core data and all metadata
- WIS2 Global Discovery Catalogues providing a search API for published discovery metadata
- WIS2 Global Monitors scraping metrics from WIS2 Global Services, and providing metrics/insights on WIS2 performance

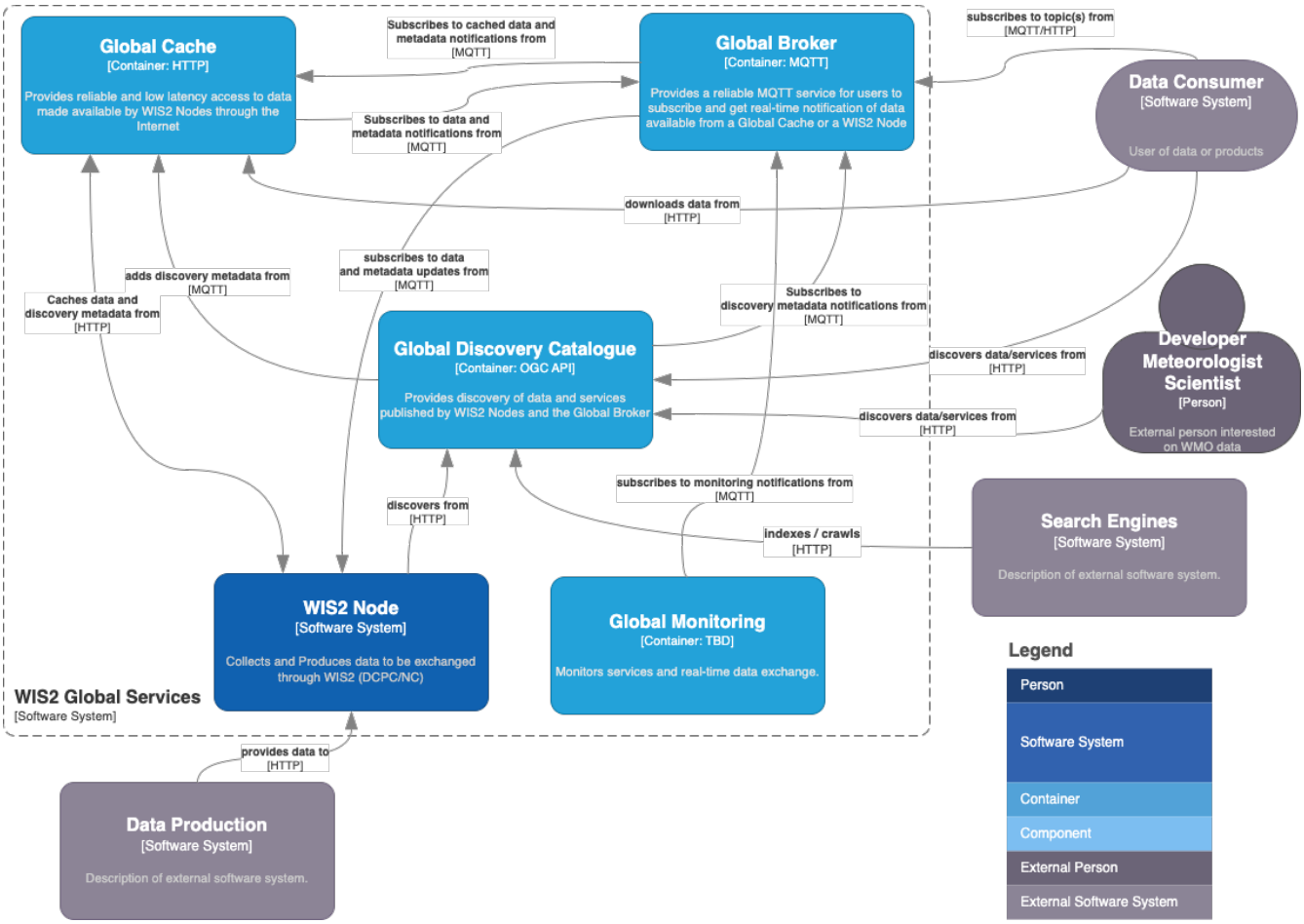

Figure 1. High Level Overview of the WIS2 Architecture

The rest of this section describes the components deployed and standards implemented as part of WIS2.

6.1. WIS2 Specifications

6.1.1. WIS2 Topic Hierarchy (WTH)

WTH defines the structure of the WIS Topic Hierarchy. Topics are utilized by WIS Nodes, Global Broker services, and data/metadata subscribers.

6.1.2. WIS2 Notification Message (WNM)

WNM defines the content, structure, and encoding for the WIS2 Notification Message Encoding. WNMs 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).

6.1.3. WMO Core Metadata Profile (WCMP2)

WCMP2 defines the content, structure, and encoding for WMO resources. WMO resources include, but are not limited to, data (NWP models, observations, forecasts and warnings, etc.), services/APIs, and processes.

6.2. WIS2 Components

6.2.1. Global Broker

WIS2 incorporates several Global Brokers, ensuring highly resilient distribution of notification messages across the globe.

6.2.2. Global Cache

A Global Cache provides a highly available data server from which a Data Consumer can download Core data, as specified in the WMO Unified Data Policy, Resolution 1 (Cg-Ext(2021)).

6.2.3. Global Discovery Catalogue

A Global Discovery Catalogue enables a data consumer to search and browse descriptions of data published by each WIS2 Node. The data description (i.e., discovery metadata) provides sufficient information to determine the usefulness of data and how one may access it.

6.2.4. Global Monitor

A Global Monitor tracks what data is published by WIS2 Nodes, whether data can be effectively accessed by Data Consumers, and the performance of components in the WIS2 system.

6.3. Testing framework

6.3.1. Data

Test data and notification messages are generated dynanamically within the testing environment. More information can be found in https://github.com/wmo-im/wis2-global-services-testing/blob/main/tools/Documentation/Activating%20the%20tests%20on%20the%20fake%20WIS2%20Nodes.adoc

Test WCMP2 records are found in https://github.com/wmo-im/wis2-global-services-testing/tree/main/tests/global_discovery_catalogue/metadata.

6.3.2. Environment

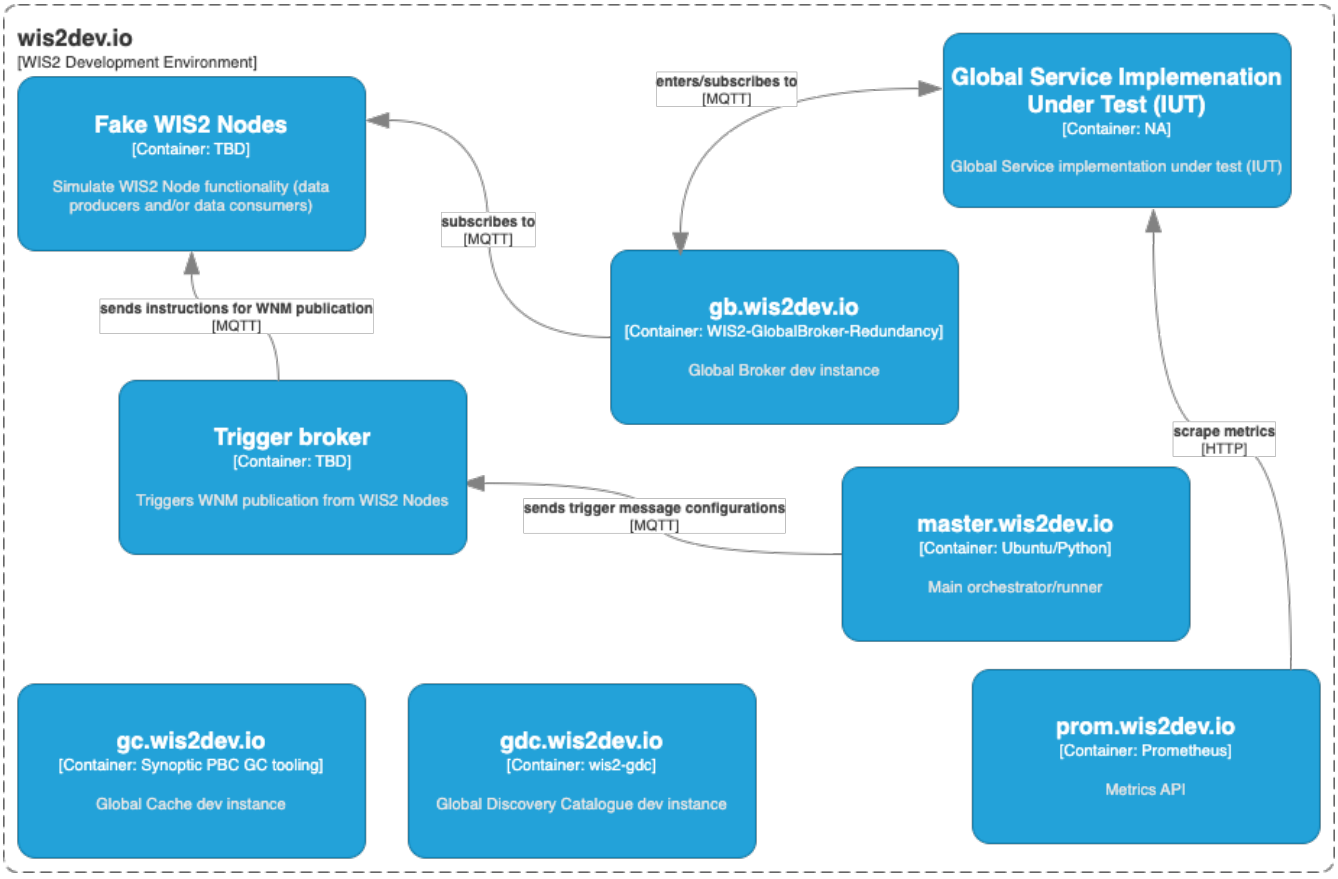

Figure 2. WIS2 Development Environment

The WIS2 development environment will be used as the target network for executing tests.

The environment to perform tests is made available on `master.wis2dev.io`. Access is coordinated with the testing team.

6.3.2.1. Setup

The following steps are required to setup the initial test environment:

```
cd /data/wis2-testing/
python3 -m venv env
. env/bin/activate
git clone https://github.com/wmo-im/wis2-global-services-testing.git
cd wis2-global-services-testing/tests
pip3 install -r requirements.txt
```

6.3.2.2. Environment variables

The testing environment includes two file for environment variables, which are initiated in the following order:

- secrets: `/data/wis2-testing/wis2-global-services-testing/tests/secrets.env`
- default: `/data/wis2-testing/wis2-global-services-testing/tests/default.env`

To set environment variables which should not be managed in Git, use secrets and refer to these environment variables in default.

See each Global Service testing section for any specific updates to environment variables.

6.3.2.3. Execution

The following steps are required prior to executing a given tests:

```
cd /data/wis2-testing/wis2-global-services-testing/tests
. /data/wis2-testing/env/bin/activate
git pull origin main
```

From here, test execution is specific to the Global Service.

6.3.2.4. Results

Test results shall be stored in `/data/wis2-testing/results`, organized by the centre id of the Global Service Implementation Under Test (IUT).

6.3.3. Functional testing

Functional testing ensures WIS2 Global Services operate with one another as expected and meet requirements.

6.3.4. Performance testing

Performace testing ensures that WIS2 Global Services are able to operate under various loads.

Chapter 7. Tests

7.1. Global Broker Service testing

All Global Services, and in particular Global Brokers and Global Caches, are collectively responsible in making the WIS a reliable and efficient mean to exchange data required for the operations of all WIS Centres. The agreed architecture provides a redundant solution where the failure of one component will not impact the overall level of service of WIS. Each Global Service should aim at achieving at least 99.5% availability of the service they propose. This is not a contractual target. It should be considered by the entity providing the Global Service as a guideline when designing and operating the Global Service.

A Global Broker:

- should support a minimum of **200** WIS2 Nodes or Global Services
- should support a minimum of **1000** subscribers.
- should support processing of a minimum of **10000** messages per second

7.1.1. Setup and tear down Global Brokers for the tests

It has been decided to rely on *-test* systems to emulate WIS2 Nodes. Two kinds of test WIS2 Nodes have been prepared.

- 10 WIS2 Nodes for functional tests
- 200 WIS2 Nodes for performance tests

7.1.1.1. For the functional tests

Ten WIS2 Nodes are deployed on AWS; The DNS names of the WIS2 nodes will be:

- test-node-1.wis2dev.io
- test-node-2.wis2dev.io

…

- test-node-10.wis2dev.io

The *centre_id* for these WIS2 Nodes will be **io-wis2dev-11-test** (on test-node-1 host) to **io-wis2dev-20-test** (on test-node-10 host).

They will be used during the functional tests of the Global Broker.

7.1.1.2. For the performance tests

Two hundred WIS2 Nodes, only for WIS2 Notification Messages, will be deployed on 5 of the same VMs. On each VM, the equivalent of 40 WIS2 Nodes will be deployed. They will share the same MQTT Broker, however the *centre_id* and the subscription topic will be different for each WIS2 Node.

The DNS name of the VM will be:

- test-node-1.wis2dev.io (for centre_id from 100 to 139)
- test-node-2.wis2dev.io (for centre_id from 140 to 179)

...

- test-node-5.wis2dev.io (for centre_id from 260 to 299)

7.1.1.3. MQTT Configuration

The MQTT broker(s) will be reachable using:

```
MQTT_SUB_BROKER=mqtt://test-node-1.wis2dev.io (and also -2, -3,... -10)
MQTT_SUB_USERNAME=everyone
MQTT_SUB_PASSWORD=onlyone
```

1. Default MQTT port will be used (1883)
2. The password is changed to *onlyone*. This allows to prevent the default authentication (everyone/everyone) to be usable, while keeping the authorization (typically using only the username) unchanged

7.1.1.4. Notification Message tests

As no metadata record will be provided, checking the existence of the metadata record by the Global Broker must be disabled.

As the topic used for the test messages will be correct, checking the validity of the Topic Hierarchy must be done.

As the Notification Messages will be syntactically correct, checking the compliance of the Notification Message with the defined schema (schemas.wmo.int) must be done.

7.1.1.5. centre_id and subscription

The centre_id will be of the from io-wis2dev-[number]-test with number varying from 11 to 20, and 100 to 299. 100 to 139 on test-node-1.wis2dev.io and so on.

Messages will be published on the topic `origin/a/wis2/io-wis2dev-x-test/#` (with x varying from 11 to 20 and 100 to 299).

7.1.1.6. Summary

Using the antiloop software used by Brazil and France, as an example, the configuration file *io-wis2dev-100-test.env* will be :

```
MQTT_SUB_BROKER=mqtt://test-node-1.wis2dev.io
MQTT_SUB_USERNAME=everyone
MQTT_SUB_PASSWORD=onlyone
```

```
MQTT_SUB_TOPIC=origin/a/wis2/io-wis2dev-100-test/#
CENTRE_ID=io-wis2dev-100-test
MSG_CHECK_OPTION=verify
TOPIC_CHECK_OPTION=verify
METADATA_CHECK_OPTION=no
GDC_URL=https://wis2-gdc.weather.gc.ca/collections/wis2-discovery-
metadata/items?lang=en&f=json&q=
SCHEMA_URL=https://raw.githubusercontent.com/wmo-im/wis2-notification-
message/main/schemas/wis2-notification-message-bundled.json
```

The configuration file *io-wis2dev-210-test.env* will be :

```
MQTT_SUB_BROKER=mqtt://test-node-3.wis2dev.io
MQTT_SUB_USERNAME=everyone
MQTT_SUB_PASSWORD=onlyone
MQTT_SUB_TOPIC=origin/a/wis2/io-wis2dev-210-test/#
CENTRE_ID=io-wis2dev-210-test
MSG_CHECK_OPTION=verify
TOPIC_CHECK_OPTION=verify
METADATA_CHECK_OPTION=no
GDC_URL=https://wis2-gdc.weather.gc.ca/collections/wis2-discovery-
metadata/items?lang=en&f=json&q=
SCHEMA_URL=https://raw.githubusercontent.com/wmo-im/wis2-notification-
message/main/schemas/wis2-notification-message-bundled.json
```

Summary of the correspondance between the MQTT endpoint address and centre_id:

- WIS2 Node 100 to 139 on test-node-1.wis2dev.io
- WIS2 Node 140 to 179 on test-node-2.wis2dev.io
- WIS2 Node 180 to 219 on test-node-3.wis2dev.io
- WIS2 Node 220 to 259 on test-node-4.wis2dev.io
- WIS2 Node 260 to 299 on test-node-5.wis2dev.io

Ahead of the testing period, each Global Broker operator is invited to prepare the configuration for the 210 (10 + 200) WIS2 Nodes.

7.1.2. Setup the Global Broker for the tests

In order to run the tests, the operators of the Global Broker will need to:

1. Remove all the subscriptions to existing and *real* WIS2 Nodes
2. Modify the password of the *everyone* user on the broker of the Global Broker from **everyone** to **onlyone**
3. Delete all existing subscriptions to broker of the Global Broker (Users, other Global Brokers, Global Caches have a subscription to the broker)
4. Enable all configurations to the *test* 210 WIS2 nodes

Depending on the Global Broker and the MQTT broker used, the method of doing the four steps above will be different and is beyond the scope of this document.

When those steps are completed, the Global Broker will be ready to run the functional and performance tests.

7.1.3. Tear down the Global Broker to reverse WIS2 operations

After the performance tests, the operators of the Global Broker will need to:

1. Remove all the subscriptions to *-test* WIS2 Nodes
2. Modify the password of the *everyone* user on the broker of the Global Broker from **onlyone** to **everyone**
3. Delete all existing subscriptions to broker of the Global Broker
4. Enable all configurations to the *real* WIS2 nodes

Depending on the Global Broker and the MQTT broker used, the method of doing the four steps above will be different and is beyond the scope of this document.

When those steps are completed, the Global Broker will be back to normal operations.

7.1.4. Functional tests

7.1.4.1. Port

Purpose

An MQTT client must be able to connect to the local broker of the Global Broker on ports 8883 (MQTTS) or 443 (WSS) using the agreed protocols with Transport Layer Security (TLS) and username/password authentication.

Requirements

- Global Broker MQTT details.
- MQTT Test Client

Steps

1. Initialise the test MQTT client with the necessary parameters such as the MQTT protocol, TLS security, and username/password for authentication (connection string).
2. Attempt to connect the MQTT broker of the Global Broker using the connection string.

Evaluate

1. Check if the connection is successful.

7.1.4.2. Certificate

Purpose

The Global Broker service must use a valid certificate. Transport Layer Security (TLS) is an encryption protocol that provides secure connections between servers and applications on the internet.

Requirements

- Global Broker MQTT connection string
- MQTT Test Client (If used, MQTT Explorer needs to import the Certificate Authority used by the Global Broker to check that the certificate is valid) or a browser like Firefox.

Steps

From the client and try to connect to a Global Broker using WSS protocol. The Global Broker sends the MQTT client its TLS certificate. The MQTT client then verifies that the certificate is valid and digitally signed by a trusted CA by comparing it with information it stores about trusted CAs. The signed certificate verifies the website server's public key, which confirms that you're communicating with the genuine server of the website you're visiting. The server also authenticates a key exchange, resulting in a one-time session key that is used to send encrypted and authenticated data between the clients and the server. If a browser like Firefox is used, connect WSS endpoint (https://globalbroker.example.org/mqtt). In the address bar, a lock is displayed.

Evaluate

1. Check if the TLS connection is successful
2. Check for certification verification.

If the connection is successful and the certificate are valid, the test passes. If the connection is not successful or the certificate is invalid, the test fails.

7.1.4.3. Origin and Cache Read-Access

Purpose

The Global Broker service must allow only read access to *origin/a/wis2/ and cache/a/wis2/* using a username and password credential of everyone/everyone

Requirements

- Global Broker MQTT connection string
- MQTT Test Client

Steps

1. From a MQTT client, set up a new connection to the Global Broker, with the following configuration settings:
2. Configure 2 subscriptions. First, create separate subscriptions for *origin/a/wis2/ and cache/a/wis2/* using a username and password credential for "everyone/everyone"

3. Save the configuration and click connect

Evaluate

Check if the connection is successful, and depending on the flow of messages, messages should appear rapidly. If messages are displayed, the test passes. If the connection is not successful, the test fails.

7.1.4.4. Deny write access to *origin/a/wis2/ and cache/a/wis2/* for everyone/everyone credentials

Purpose

The Global Broker service must prevent write access to any topic with everyone/everyone credentials

Requirements

- Global Broker MQTT connection string
- MQTT Test Client

Steps

1. Use an MQTT client to connect to Global Broker
2. Try to publish data or metadata to Global Broker

Evaluate

Check if the connection is successful, and the publication fails or the connection drops, the test is successful. If the connection is successful, and the publication is allowed, the test fails.

7.1.4.5. cluster redundancy

Purpose

The Global Broker service, should be using a MQTT server deployed in a cluster, then the MQTT Broker must use a redundant load balancing service so that the service in maintained in case of failure of one entity of the cluster

Requirements

- Global Broker MQTT connection string
- MQTT Test Client

Steps

1. From a MQTT client, set up a new subscription to either "origin/a/wis2/" and "cache/a/wis2/" using a username and password credential for "everyone/everyone".
2. Fail a member of the cluster and ensure that subscriptions are still being fulfilled

Evaluate

1. Check if the subscription is successful even after the members of the cluster are failed. If the subscription continues as cluster is altered, the test passes. If the subscription is not fulfilled after cluster alternation, the test fails.

7.1.4.6. Discarding of duplicate messages

Purpose

The Global Broker service must discard all duplicated messages (identical id) received whatever the originator of the messages

Requirements

- The container https://hub.docker.com/r/golfvert/fakewis2node deployed on 10 tests system (**io-wis2dev-11-test** to **io-wis2dev-20-test**)
- Global Broker MQTT connection string to 2 WIS2 Nodes (with the following centre_id: io-wis2dev-11-test and io-wis2dev-12-test)
- MQTT Test Client subscribed to **origin/a/wis2/io-wis2dev-11-test/#** and **origin/a/wis2/io-wis2dev-20-test/**

Steps

1. WIS2Node *io-wis2dev-11-test* publish on its local broker **ten** messages with a pre-defined id (using the UUID format) on topic **origin/a/wis2/io-wis2dev-11-test/core/data/weather/surface-based-observation/synop**
2. WIS2Node *io-wis2dev-12-test* publish **ten** message with the same id (same id as above) on topic **origin/a/io-wis2dev-12-test/core/data/weather/surface-based-observation/synop**

Evaluate

1. If the Global Broker discards all messages except one, makes it available on one of the two topics depending the WIS2 Node messages that arrived first.
2. The MQTT client received one message
3. Increments **wmo_wis2_gb_messages_subscribed_total** by 10 on both centre_id
4. Increments **wmo_wis2_gb_messages_published_total** by 1 on centre_id from the WIS2Node that arrives first (io-wis2dev-11-test or io-wis2dev-12-test)
5. If both statements are true, the test passes. Otherwise, the test fails.

7.1.4.7. Publishing a message using the centre_id from a different WIS2 Node

Purpose

The Global Broker service must ensure that any WIS2 Node is not publishing a message using a centre_id from another WIS2 Node

Requirements

- The container https://hub.docker.com/r/golfvert/fakewis2node deployed on 10 tests system (**io-wis2dev-11-test** to **io-wis2dev-20-test**)
- Global Broker MQTT connection string to **io-wis2dev-11-test**
- MQTT Test Client

Steps

1. Have WIS2Node *io-wis2dev-11-test* publish a valid message on topic Eg

Evaluate

1. The Global Broker ignores (in fact the message will not be received at all) the message published. No metrics is incremented
2. If the message is received by MQTT Client then the test fails.

7.1.4.8. Publishing messages from a WIS2 Node using valid topics (compliant with WIS2 Topic Hierarchy)

Purpose

The Global Broker service must forward messages when the topic is compliant with the WIS2 Topic Hierarchy

Requirements

- The container https://hub.docker.com/r/golfvert/fakewis2node deployed on 5 tests system (**io-wis2dev-11-test** to **io-wis2dev-20-test**)
- Global Broker MQTT connection string to a WIS2 Nodes (**io-wis2dev-11-test**)
- MQTT Test Client

Steps

1. Have WIS2Node **io-wis2dev-11-test** publish **valid** messages on **valid** topics hierarchy (one message per tested valid topic)

Evaluate

1. The Global Broker forward all messages
2. The MQTT client receives all messages
3. Increments **wmo_wis2_gb_messages_subscribed_total** by 1 for each message
4. Increments **wmo_wis2_gb_messages_published_total** by 1 for each message
5. If all above statements are true, the test passes. Otherwise, the test fails.

7.1.4.9. Publishing messages from a WIS2 Node using invalid topics (not compliant with WIS2 Topic Hierarchy)

Purpose

The Global Broker service must forward messages when the topic is compliant with the WIS2 Topic Hierarchy

Requirements

- The container https://hub.docker.com/r/golfvert/fakewis2node deployed on 5 tests system (**io-wis2dev-11-test** to **io-wis2dev-20-test**)
- Global Broker MQTT connection string to a WIS2 Nodes (**io-wis2dev-11-test**)
- MQTT Test Client

Steps

1. Have WIS2 Node **io-wis2dev-11-test** publish 10 **valid** messages on 10 **different** and **invalid** topics

Evaluate

1. The Global Broker discards all messages
2. The MQTT client doesn't receive any message
3. Increments **wmo_wis2_gb_messages_subscribed_total** by 10
4. Increments **wmo_wis2_gb_messages_invalid_topic_total** by 10
5. If all above statements are true, the test passes. Otherwise, the test fails.

7.1.4.10. Publishes messages from a WIS2 Node on a *valid* topic without corresponding metadata

Purpose

The Global Broker service must check that the topic used to publish a message by a WIS2 Node is announcing the availability of data with corresponding metadata.

Requirements

- The container https://hub.docker.com/r/golfvert/fakewis2node deployed on 5 tests system (**io-wis2dev-11-test** to **io-wis2dev-20-test**)
- Global Broker MQTT connection string to a WIS2 Nodes (**io-wis2dev-11-test**)
- MQTT Test Client

Steps

1. Have WIS2 Node **io-wis2dev-11-test** publish 10 **valid** messages on 10 **different** topics, and no metadata exists for any topic used.
2. e.g Publish on **origin/a/wis2/*io-wis2dev-11-test*/core/data/weather/surface-based-**

observation/synop and WIS2 Node **io-wis2dev-11-test** has not published a metadata record for the synop.

Evaluate

1. The Global Broker discards all messages
2. The MQTT client doesn't receive any message
3. Increments **wmo_wis2_gb_messages_subscribed_total** by 10
4. Increments **wmo_wis2_gb_messages_no_metadata_total** by 10
5. If all above statements are true, the test passes. Otherwise, the test fails.

7.1.4.11. Verifying the compliance of a WIS2 Notification message

Purpose

The Global Broker service must verify the compliance of the WIS2 Notification Message with the agreed standard as specified in the Manual on WIS Vol. 2

Requirements

1. The container https://hub.docker.com/r/golfvert/fakewis2node deployed on 5 tests system (**io-wis2dev-11-test** to **io-wis2dev-20-test**)
2. Global Broker MQTT connection string to a WIS2 Nodes (**io-wis2dev-11-test**)
3. MQTT Test Client

Steps

1. Have WIS2 Node **io-wis2dev-11-test** publish 10 **invalid** messages on 10 **valid** topics
2. All mandatory fields of the WIS2 Notification Messages must be tested

Evaluate

1. The Global Broker discards all messages
2. The MQTT client doesn't receive any message
3. Increments **wmo_wis2_gb_messages_subscribed_total** by 10
4. Increments **wmo_wis2_gb_messages_invalid_messages_total** by 10
5. If all above statements are true, the test passes. Otherwise, the test fails.

7.1.5. Performance tests

We must ensure that the Global Broker service performs properly under stress. The following outlined tests will test the Global Broker service prior to transition of WIS2 to an operational state on January 1, 2025

7.1.5.1. Minimum number of WIS2 Nodes

Purpose

The Global Broker service should support a minimum of **200** WIS2 Nodes

Requirements

1. The container https://hub.docker.com/r/golfvert/benchmarkwis2gb deployed on 5 tests system
2. Global Broker subscribing to **200** WIS2 Nodes (**io-wis2dev-100-test** to **io-wis2dev-299-test**)
3. MQTT Test Client

Steps

1. On each of the 200 WIS2 Nodes, publish 10 **valid** messages, on **valid** topic, without associated metadata, and with different **id** messages
2. On the MQTT test client, subscribe to **origin/a/wis2/#**

Evaluate

1. If on the MQTT test client, 10 messages for each of the 200 centre-id are received, the test passes. Otherwise, it fails.

7.1.5.2. Minimum number of subscribers

Purpose

The Global Broker service should support a minimum of **1000** subscribers.

Requirements

- Global Broker
- MQTTX CLI (https://mqttx.app/docs/cli) deployed on 5 tests systems

Steps

1. Use MQTTX CLI *bench* on each test system to simulate 200 clients by using `mqttx bench conn -c 200 -i 100` and the relevant connection information for the Global Broker being tested.

Evaluate

1. If the output of the command on each test system is similar to:

```
mqttx bench conn -c 200 -i 100
  Starting connect benchmark, connections: 200, req interval: 100ms
✔  [200/200] - Connected
✔  Created 200 connections in 22.355s
```

with 200 connections created, on the 5 tests systems, this test is successful.

7.1.5.3. Minimum number of messages per second

Purpose

The Global Broker service should support processing of a minimum of **10000** messages per second.

Requirements

1. The container https://hub.docker.com/r/golfvert/benchmarkwis2gb deployed on 5 tests system (different from above)
2. Global Broker subscribing to **200** WIS2 Nodes (**io-wis2dev-100-test** to **io-wis2dev-299-test**)
3. MQTTX CLI deployed on 5 tests systems.

Steps

1. On each of the 200 WIS2 Nodes, publish X **valid** messages per second during Y seconds, on a **valid** topic, without associated metadata or data, and with different **id** messages
2. Use MQTTX CLI *bench* on each test VMs to simulate Z clients by using `mqttx bench sub -c Z -t origin/a/wis2/#` and the relevant connection information for the Global Broker being tested.

Typically with:

1. With X = 5 messages per second on each of the 200 WIS2 Nodes, this will create 1000 messages per second
2. With Z = 2 on each VM, this will create 10 subscriptions.
3. As every subscriber will get all messages will be equivalent to **10000** messages per second.

Evaluate

1. Run the test for 30 seconds (Y=30), if MQTTS CLI output shows that 300000 are received, then the test is successful.

7.2. Global Cache Service Testing

7.2.1. Preparation

7.2.1.1. Setup

Before running the pytest tests, ensure the following setup steps are completed:

Disconnect the Global Cache from the production environment.

```
Ensure that the Global Cache is not connected to the production environment during the
testing session. This is to prevent any test messages from being propagated to the
production environment.
```

1. Change the username and password for the GC's local MQTT broker to the dev/test credentials.

This ensures that no test messages will be propagated to the production environment.

- The username and password for the local MQTT broker of the Global Cache should be set to the dev/test credentials.
- The connection string to be used is: `mqtts://everyone-dev:everyone-dev@your-gc-mqtt-broker-host:8883`

2. Remove other subscriptions to prod GB's.
   - Ensure that the Global Cache is not subscribed to any other Global Broker except the dev/test Global Broker.

Connect the Global Cache to the dev/test environment.

```
Ensure that the Global Cache is connected to the dev/test environment during the
testing session. This is to allow the test messages to be propagated to the Global
Cache for testing.
```

1. GB Subscription to the GC's Local MQTT Broker.
   - Ensure the dev/test Global Broker is subscribed to the GC's local MQTT broker. A connection string must be provided prior to the start of the testing session.
   - This is the same connection string mentioned above:
     - `mqtts://everyone-dev:everyone-dev@your-gc-mqtt-broker-host:8883`
   - This step will be taken care of by the test administrator assuming the connection string is provided.
2. GC Subscription to the dev/test Global Broker.
   - Ensure the Global Cache is subscribed to the dev/test Global Broker (`gb.wis2dev.io`).
     - the connection string to be used is: `mqtts://everyone-dev:everyone-dev@gb.wis2dev.io:8883`
   - The Global Cache should be subscribed to the following topics:
     - `origin/a/wis2/+/data/#`
     - `cache/a/wis2/+/data/#`
     - `origin/a/wis2/+/metadata/#`
     - `cache/a/wis2/+/metadata/#`

pytest setup

```
Prior to executing the tests, update the global-cache.env file with the necessary
environment variables. This env file is nested in the tests/global_cache directory and
the values of the following variable must be updated:
```

- `GC_METRICS_REPORT_BY` - this is the centre_id of the global service and matches the report_by property in the metrics.

- `TEST_GC_MQTT_BROKER` - this is the MQTT broker host for the Global Cache being tested. This should be the same connection string that was provided to the test administrators to configure the dev/test Global Broker to subscribe to the GC's local MQTT broker.

7.2.1.2. Teardown

To teardown the configuration after the testing session, simply reverse the setup steps:

- Unsubscribe the Global Cache from the dev/test Global Broker.
- Unsubscribe the dev/test Global Broker from the GC's local MQTT broker.
- Reset the metrics so that dev and prod metrics are not mixed.
- Change the username and password for the GC's local MQTT broker back to the production credentials.
- Reconnect the Global Cache to the production environment by re-subscribing to the production Global Brokers.

7.2.2. Functional Tests

7.2.2.1. MQTT Broker Connectivity

Purpose

An MQTT client must be able to connect to the local broker of the Global Cache on port 8883 using the MQTT protocol version 5 with TLS (i.e., mqtts protocol) and username/password authentication.

Source: Manual on WIS (WMO No. 1060), Vol II, clause 3.7.5.2: A Global Cache shall operate a message broker.

Requirements

- GC MQTT broker connection string
- MQTT Test Client

Steps

1. Initialize the test MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication (connection string).
2. Attempt to connect the MQTT broker of the Global Cache using the connection string.

Evaluate

1. Check if the connection is successful (rc code). If the connection is successful, the test passes. If the connection is not successful, the test fails.

7.2.2.2. GC MQTT Broker Subscription

Purpose

A Global Cache must allow connected MQTT clients to subscribe to the cache/a/wis2/# topic using a provided connection string.

Requirements

- GC MQTT broker connection string
- MQTT Test Client

Steps

1. Initialize a MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication (connection string).
2. Connect the MQTT client to the local broker of the Global Cache.
3. Once the connection is successful, attempt to subscribe to the `cache/a/wis2/#` topic.

Evaluate

1. Check if the subscription is successful. If the subscription is successful based on the returned rc code (SUBACK), the test passes. If the subscription is not successful, the test fails.
2. Close the connection to the broker after the test.

7.2.2.3. WIS2 Notification Message (WNM) Processing

Purpose

Test that the GC functions as expected under normal conditions. The Global Cache should process a valid incoming WNM, download the data at the provided canonical link, and publish a new WNM on the proper cache/ topic using the proper message structure, and update the necessary GC metrics.

This test also evaluates the client data download requirement: An HTTP client (i.e., a Web browser) must be able to connect to the HTTP server of the Global Cache on port 443 using HTTP 1.1 with TLS but without any authentication and be able to resolve the URL provided in a data download link (a link object's `href` property where `rel=canonical`) from a notification message published by the Global Cache within the previous 24 hours; i.e., download a cached data item.

Source: Manual on WIS (WMO No. 1060), Vol II, clause 3.7.5.5: A Global Cache shall provide highly available access to copies of discovery metadata records and core data it stores; clause 3.7.5.6: A Global Cache shall retain a copy of the discovery metadata records and core data it stores for a duration compatible with the real-time or near-real-time schedule of the data and not less than 24 hours; clause 4.5.2: A Global Cache shall download core data and discovery metadata from [WIS2 Nodes] and other Global [Services] to provide for reliable, low-latency access to those resources via WIS; clause 4.5.6: Data and discovery metadata available for download from a Global Cache shall be accessible via a URL using at least one of the protocols specified [...].

Source: Manual on WIS (WMO No. 1060), Vol II, clause 3.7.5.4: Based on the notifications it receives, a Global Cache shall download and store a copy of discovery metadata records and core data from

[WIS2 Nodes] and other Global [Services]; clause 3.7.5.7: A Global Cache shall publish notifications via its Message Broker about copies of the discovery metadata records and core data it makes available. A Global Cache shall use a standardized topic structure when publishing notifications; clause 4.5.2: A Global Cache shall download core data and discovery metadata from [WIS2 Nodes] and other Global [Services] to provide for reliable, low-latency access to those resources via WIS; clause 4.5.4: Based on received notifications, a Global Cache shall download core data from [WIS2 Nodes] or other Global [Services] and store them for a minimum duration of 24 hours; clause 4.5.5: Based on its received notifications, a Global Cache shall download discovery metadata records from [WIS2 Nodes] or other Global [Services] and store them for a minimum duration of 24 hours; clause 4.5.7: A Global Cache shall publish notifications to a Message Broker indicating the availability of data and discovery metadata resources from the Global Cache and shall use the format and protocol specified [...].

Source: Guide to WIS (WMO No. 1061), Vol II, clause 2.7.4.1. [Global Cache] Technical considerations https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html#_technical_considerations_2; clause 2.7.4.2. [Global Cache] Practices and procedures https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html#_practices_and_procedures_2

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - origin/a/wis2/+/data/#
  - cache/a/wis2/+/data/#
  - origin/a/wis2/+/metadata/#
  - cache/a/wis2/+/metadata/#
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- Prepared WIS2 Notification Messages and associated data objects:
  - A known number **valid** WNM's with:
    - `properties.cache` set to true
    - `properties.data_id` + `properties.pubtime` should be unique to each message. Ensuring a different data_id is best here.
  - Accompanying data objects should be accessible via the canonical link provided in the WNM.
    - The canonical link should be accessible per the core requirements and the data object hash should match the hash provided in the WNM if integrity properties are provided.

Steps

1. Configure the MQTT test client to connect to the dev/test GB MQTT broker using the provided connection string.
2. Publish a batch of Prepared WIS2 Notification Messages to the dev/test GB on following topics:
    - Send 1 or more messages to origin/a/wis2/+/data/#
    - Send 1 or more messages to cache/a/wis2/+/data/#
    - Send 1 or more messages to origin/a/wis2/+/metadata/#
    - Send 1 or more messages to cache/a/wis2/+/metadata/#
3. The test MQTT client should store the messages received on the `cache/a/wis2/#` topic published by the GC and download the data objects from the canonical link provided in the messages using HTTP 1.1 with TLS.
    - The original data object and the downloaded>>cached data objects can then be compared to ensure they are identical.

Evaluate

- WNM Messages
    - The total number of cache notification messages published by the GC on the cache/a/wis2/# topic.
    - All messages should be the same as the source WNM's except for:
        - The canonical link (a link object's `href` property where `rel=canonical`), this should point to the GC's cached object.
        - the unique identifier of the message (id)
        - The topic, always on the `cache` channel. Note the incoming message may be unchanged if it was originally published on the `cache` channel.
- Data Objects
    - The total number of data objects cached by the GC. This should match the number of cache notification messages published.
    - The data objects cached by the GC should be identical to the source data objects.
        - The diff or hashes of the data objects should be identical.
- GC Metrics
    - `wmo_wis2_gc_download_total` (matches total messages)
    - `wmo_wis2_gc_dataserver_status_flag` (set to 1 for each)
    - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (set for each and within expected time range)

7.2.2.4. Cache False Directive

Purpose

Where a Global Cache receives a notification message with *properties.cache* set to false, the Global Cache should publish a notification message where the data download link (a link object's `href` property where `rel=canonical`) refers to the source data server.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated with subscription to the `cache/a/wis2/#` topic and `origin/a/wis2/#` topic of the dev/test GB.
- MQTT test client
  - Client should connect to both the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- Prepared WIS2 Notification Messages and data objects:
  - A known number **valid** WNM's with:
    - `properties.cache` set to false
    - `properties.data_id` + `properties.pubtime` should be unique to each message.
  - Accompanying data objects are not required for this test.

Steps

1. Configure the MQTT test client to connect to the dev/test GB MQTT broker using the provided connection string.
2. Publish the prepared WIS2 Notification Messages to the dev/test GB the following topics:
   - Send 1 or more messages to origin/a/wis2/+/data/#
   - Send 1 or more messages to cache/a/wis2/+/data/#
   - Send 1 or more messages to origin/a/wis2/+/metadata/#
   - Send 1 or more messages to cache/a/wis2/+/metadata/#

Evaluate

- WNM Messages
  - The total number of cache notification messages published by the GC on the `cache/a/wis2/#` topic
  - all messages should be the same as the source WNM's except for:
    - the unique identifier of the message (id)
    - the topic (`cache/a/wis2/⋯`) (note the incoming message may be on the same `cache/#` topic if it is from another GC)

- GC Metrics
  - `wmo_wis2_gc_download_total` (unchanged)
  - `wmo_wis2_gc_dataserver_status_flag` (unchanged)
  - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (unchanged)
  - `wmo_wis2_gc_no_cache_total` (+=1 for each WNM)

7.2.2.5. Source Download Failure

Purpose

Where a Global Cache receives a valid WNM, but is unable to download a data item from the location specified in a notification message (i.e., the source data server), the metric `wmo_wis2_gc_dataserver_status_flag` for the source data server should be set to 0 (zero).

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated with subscription to the `cache/a/wis2/#` topic and `origin/a/wis2/#` topic of the dev/test GB.
- MQTT test client
  - Client should connect the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number **valid** WNM's with:
    - invalid data download links (a link object's `href` property where `rel=canonical`)
    - `properties.data_id` + `properties.pubtime` should be unique to each message.
  - Accompanying data objects are not required for this test.

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WNM's to the dev/test GB on one or more of the following topics:
   - origin/a/wis2/+/data/#
   - cache/a/wis2/+/data/#
   - origin/a/wis2/+/metadata/#
   - cache/a/wis2/+/metadata/#

Evaluate

- WNM Messages
  - No messages should be published on the `cache/a/wis2/#` topic as received by the test MQTT client.
- Data Objects
  - No data objects should be cached by the GC.
- GC Metrics
  - `wmo_wis2_gc_download_total` (unchanged)
  - `wmo_wis2_gc_dataserver_status_flag` (set to 0 for each)
  - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (unchanged)
  - `wmo_wis2_gc_downloaded_errors_total` (+=1 for each WNM)

7.2.2.6. Data Integrity Check Failure

Purpose

A Global Cache should validate the integrity of the resources it caches and only accept data which matches the integrity value from the WIS Notification Message. If the WIS Notification Message does not contain an integrity value, a Global Cache should accept the data as valid. In this case a Global Cache *may* add an integrity value to the message it republishes.

Source: Guide to WIS (WMO No. 1061), Vol II, clause 2.7.4.1. [Global Cache] Technical considerations https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html#_technical_considerations_2; clause 2.7.4.2. [Global Cache] Practices and procedures https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html#_practices_and_procedures_2 **Source:** https://github.com/wmo-im/wis2-notification-message/blob/main/standard/recommendations/core/REC_integrity.adoc

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated with subscription to the `cache/a/wis2/#` topic and `origin/a/wis2/#` topic of the dev/test GB.
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number **valid** WNM's with:
    - invalid data integrity value (accessed via `properties.integrity.value` and the method

specified in `properties.integrity.method`)

- `properties.data_id` + `properties.pubtime` should be unique to each message.

- Accompanying data objects that are accessible via the canonical link provided in the WNM

Steps

1. Publish the prepared WNM's to the dev/test GB on one or more of the following topics:
   - origin/a/wis2/+/data/#
   - cache/a/wis2/+/data/#
   - origin/a/wis2/+/metadata/#
   - cache/a/wis2/+/metadata/#

Evaluate

- WNM Messages
  - No messages should be published on the `cache/a/wis2/#` topic as received by the test MQTT client.
- Data Objects
  - No data objects should be cached by the GC.
- GC Metrics
  - `wmo_wis2_gc_download_total` (unchanged)
  - `wmo_wis2_gc_dataserver_status_flag` (set to 0 for each)
  - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (unchanged)
  - `wmo_wis2_gc_downloaded_errors_total` (+=1 for each WNM)
  - `wmo_wis2_gc_integrity_failed_total` (+=1 for each WNM)

7.2.2.7. WIS2 Notification Message Deduplication

Purpose

A Global Cache must ensure that only one instance of a notification message with a given unique identifier (id) is successfully processed.

Source: Manual on WIS (WMO No. 1060), Vol II, clause 3.7.5.3: A Global Cache shall subscribe to notifications about the availability of discovery metadata records and core data for real-time or near-real-time exchange. Duplicate notifications are discarded.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated with subscription to the `cache/a/wis2/#` topic and `origin/a/wis2/#` topic

of the dev/test GB.

- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number **valid** WNM's where:
    - `properties.data_id` + `properties.pubtime` are NOT unique to each message, but shared by 2 or more messages.
  - Accompanying data objects that are accessible via the canonical link provided in the WNM,

Steps

1. Publish the prepared WNM's to the dev/test GB on one or more of the following topics:
   - origin/a/wis2/+/data/#
   - cache/a/wis2/+/data/#
   - origin/a/wis2/+/metadata/#
   - cache/a/wis2/+/metadata/#

Evaluate

- WNM Messages
  - Only one message should be published by the GC on the `cache/a/wis2/#` topic per unique identifier which is defined as `properties.data_id` + `properties.pubtime`.
    - Note that due to the update directive related to 8.2, prepared messages should use unique data_id's to ensure uniqueness.
- Data Objects
  - Only one data object should be cached per unique identifier which is defined as `properties.data_id` + `properties.pubtime`.
- GC Metrics
  - `wmo_wis2_gc_download_total` (+=1 for each unique identifier)
  - `wmo_wis2_gc_dataserver_status_flag` (set to 1 for each unique identifier)
  - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (set to current for each unique identifier)
  - `wmo_wis2_gc_downloaded_errors_total` (unchanged)
  - `wmo_wis2_gc_integrity_failed_total` (unchanged)

7.2.2.8. WIS2 Notification Message Deduplication (Alternative 1)

Purpose

Where a Global Cache fails to process a notification message relating to a given unique data object (`properties.data_id` + `properties.pubtime`), a Global Cache should successfully process a valid, subsequently received notification message with the same unique data identifier.

Source: Manual on WIS (WMO No. 1060), Vol II, clause 3.7.5.3: A Global Cache shall subscribe to notifications about the availability of discovery metadata records and core data for real-time or near-real-time exchange. Duplicate notifications are discarded.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated with subscription to the `cache/a/wis2/#` topic and `origin/a/wis2/#` topic of the dev/test GB.
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number **valid** WNM's where:
    - `properties.data_id` + `properties.pubtime` are NOT unique to each message, but shared by 2 or more messages.
    - This defines a unique identifier message set.
    - For each unique identifier message set, the first published message should be invalid, or the data object inaccessible, and the second message/data object should be valid.
  - Accompanying data objects that are accessible (or not) via the canonical link provided in the WNM.

Steps

1. Publish the prepared WNM's to the dev/test GB such that the invalid WNM for each unique data identifier is published first. One or more of the following topics can be used:
   - origin/a/wis2/+/data/#
   - cache/a/wis2/+/data/#
   - origin/a/wis2/+/metadata/#
   - cache/a/wis2/+/metadata/#

Evaluate

- WNM Messages
  - Only one message should be received on the `cache/a/wis2/#` topic per unique identifier

which is defined as `properties.data_id` + `properties.pubtime`.

- Data Objects
  - Only one data object should be cached per unique identifier which is defined as `properties.data_id` + `properties.pubtime`.
- GC Metrics
  - `wmo_wis2_gc_download_total` (+=1 for each unique identifier)
  - `wmo_wis2_gc_dataserver_status_flag` (set to 1 for each unique identifier)
  - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (set to current for each unique identifier)
  - `wmo_wis2_gc_downloaded_errors_total` (+=1 for each unique identifier WNM message set)
  - `wmo_wis2_gc_integrity_failed_total` (unchanged)

7.2.2.9. WIS2 Notification Message Deduplication (Alternative 2)

Purpose

Related to the two previous tests, a GC should not process and cache a data item if it has already processed and cached a data item with the same `properties.data_id` and a `properties.pubtime` that is equal to or less than the `properties.pubtime` of the new data item. This test is an extension of the previous tests and can be conducted in conjunction with them.

Requirements

See above.

Steps

1. Publish the prepared WNM's to the dev/test GB such for each unique identifier message set, the first published message has a pubtime that is greater than or equal to the subsequent message/s. One or more of the following topics can be used:
   - origin/a/wis2/+/data/#
   - cache/a/wis2/+/data/#
   - origin/a/wis2/+/metadata/#
   - cache/a/wis2/+/metadata/#

Evaluate

- WNM Messages
  - For each message set with a shared data_id, each message should be processed by the GC and received on the `cache/a/wis2/#` topic assuming that the `properties.pubtime` as been correctly set (decreasing or equal) for each message sent in chronological order.
- Data Objects
  - For each message set with a shared data_id, each data object should be cached by the GC and assuming that the `properties.pubtime` as been correctly set (decreasing or equal) for each

message sent in chronological order.

- GC Metrics
  - `wmo_wis2_gc_download_total` (+=1 for each set of messages sharing the same data_id)
  - `wmo_wis2_gc_dataserver_status_flag` (set to 1)
  - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (set to current)
  - `wmo_wis2_gc_downloaded_errors_total` (unchanged)
  - `wmo_wis2_gc_integrity_failed_total` (unchanged)

7.2.2.10. Data Update

Purpose

A Global Cache should treat notification messages with the same data item identifier (`properties.data_id`), but different publication times (`properties.pubtime`) as unique data items. Data items with the same `properties.data_id` but a greater/later publication time AND a update link (links['rel']='update'), should be processed (see test Notification processing). Data items with the same `properties.data_id` but earlier or identical publication times should be ignored (see deduplication test 8).

Source: Guide to WIS (WMO No. 1061), Vol II, clause 2.7.4.2. [Global Cache] Practices and procedures: "Verify if the message points to new or updated data by comparing the pubtime value of the notification message with the list of data_ids". https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html#_practices_and_procedures_2

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated with subscription to the `cache/a/wis2/#` topic and `origin/a/wis2/#` topic of the dev/test GB.
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number **valid** WNM's where:
    - `properties.data_id` + `properties.pubtime` are unique to each message, but the properties.data_id is shared by 2 or more messages and the pubtimes are different.
    - Ensure that for a given shared data_id, the message with the latest pubtime has link with `rel=update`.
    - This defines a unique identifier message set.

- Accompanying data objects that are accessible via the canonical link provided in the WNM.

Steps

1. Publish the prepared WNM's to the dev/test GB such for each unique identifier message set, the first published message has a pubtime that is less than the subsequent message/s and subsequent messages have a valid update link. One or more of the following topics can be used:
   - origin/a/wis2/+/data/#
   - cache/a/wis2/+/data/#
   - origin/a/wis2/+/metadata/#
   - cache/a/wis2/+/metadata/#

Evaluate

- WNM Messages
  - For each message set with a shared data_id, each message should be processed by the GC and received on the `cache/a/wis2/#` topic assuming that the `properties.pubtime` as been correctly set (increasing) for each message sent in chronological order.
- Data Objects
  - For each message set with a shared data_id, each data object should be cached by the GC and assuming that the `properties.pubtime` as been correctly set (increasing) for each message sent in chronological order.
- GC Metrics
  - `wmo_wis2_gc_download_total` (+=1 for each message)
  - `wmo_wis2_gc_dataserver_status_flag` (set to 1)
  - `wmo_wis2_gc_dataserver_last_download_timestamp_seconds` (set to current)
  - `wmo_wis2_gc_downloaded_errors_total` (unchanged)
  - `wmo_wis2_gc_integrity_failed_total` (unchanged)

7.2.3. Performance tests

7.2.3.1. WIS2 Notification Processing Rate

Purpose

A Global Cache shall be able to successfully process, on average, 2000 unique WNM's per minute with an average message size of 85kb. This test represents the average message size of the current WNM's. The noted WNM's/minute rate can be used as a performance indicator for the GC being tested.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#`

topics.

- Dev/test GC is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - origin/a/wis2/+/data/#
  - cache/a/wis2/+/data/#
  - origin/a/wis2/+/metadata/#
  - cache/a/wis2/+/metadata/#
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- GC metrics scraper
- WIS2 Notification Messages and associated data objects:
  - A known number **valid** WNM's with:
    - `properties.cache` set to true
    - `properties.data_id` + `properties.pubtime` should be unique to each message. The ensure consistency, data_id alone should be used to determine uniqueness.
  - Accompanying data objects should be accessible via the canonical link provided in the WNM.
    - The canonical link should be accessible per the core requirements and the data object hash should match the hash provided in the WNM if integrity properties are provided.
    - Average message size should be 85kb.

Steps

- Initialize the trigger client and publish the WNM dataset configuration to the dev/test GB on the topic `config/a/wis2/gc_performance_test`.
- Collect for the origin and cache messages.
- Assert that the number of origin and cache messages received is greater than 0.
- Calculate and print the following processing metrics: — Total processing time from the first origin message to the last cache message.
  a. Average processing time per message.
  b. Throughput in messages per second.
  c. Cache processing time excluding initial lag.
  d. Average cache message size.

Evaluate

- WNM Messages
  - The total number of cache notification messages published by the GC on the cache/a/wis2/# topic should match what was published (2000).

- GC Metrics
  - `wmo_wis2_gc_download_total` matches total expected messages.
- The time taken to process the messages should not exceed 60 seconds (plus time taken to publish the WNM's) in order to pass the test.
  - The results can be used as a baseline for the GC's performance.

7.2.3.2. Concurrent client downloads

Purpose

A Global Cache should support a minimum of 1000 simultaneous downloads.

Source: Manual on WIS (WMO No. 1060), Vol II, clause 3.7.5.5: A Global Cache shall provide highly available access to copies of discovery metadata records and core data it stores; clause 4.5.1: A Global Cache shall operate a highly available storage and download service; clause 4.5.2: A Global Cache shall download core data and discovery metadata from [WIS2 Nodes] and other Global [Services] to provide for reliable, low-latency access to those resources via WIS. **Source:** Guide to WIS (WMO No. 1061), Vol II, clause 2.7.2.2. Service levels, performance indicators and fair-usage policies: https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html#_procedure_for_registration_of_a_new_global_service

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on the `origin/a/wis2/#` and `cache/a/wis2/#` topics.
- Dev/test GC is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - origin/a/wis2/+/data/#
  - cache/a/wis2/+/data/#
  - origin/a/wis2/+/metadata/#
  - cache/a/wis2/+/metadata/#
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- WIS2 Notification Messages and associated data objects:
  - A known number (1) **valid** WNM with:
    - `properties.cache` set to true
    - `properties.data_id` + `properties.pubtime` should be unique to each message. Ensuring a different data_id is best here.
  - Valid data objects to be cached
    - A larger than average data object should be generated/used in order to ensure that the

clients downloading the data object concurrently do not finish before the test is complete. A 200MB data object will be used.

- ApacheBench (ab) to manage the concurrent downloads.

Steps

1. Publish the prepared WNM to the dev/test GB on one of the following topics:
   - origin/a/wis2/+/data/#
   - cache/a/wis2/+/data/#
   - origin/a/wis2/+/metadata/#
   - cache/a/wis2/+/metadata/#
2. For the WNM:
   - Receive the *cache* notification message from the dev/test GC.
   - Extract the canonical link from the cached messages.
   - Prepare and publish an 'ab' scenario configuration using the canonical link.
   - Start 1000 concurrent downloads of the data object from the canonical link, distributed across 10 test nodes.
   - Record the number of successful downloads and the time taken to complete each download.

Evaluate

The test is considered successful if the following conditions are met:

- The total number of successful downloads is 1000.
- While the download time can be used to establish a baseline, it is highly dependent on the network and server conditions of the test environment and should not be used as a pass/fail criteria.

7.2.3.3. Implicit tests

```
These are tests that are to be verified by the individual implementations as they
represent critical requirements but would be difficult to test in a generic way.
```

Valid TLS/SSL certificate

- A Global Cache must have a valid TLS/SSL certificate to ensure secure communication with other WIS2 components.

Available Storage Space

- A Global Cache shall be able to store at least 100GB of Core data items.

Source: Guide to WIS (WMO No. 1061), Vol II, clause 2.7.2.2. Service levels, performance indicators and fair-usage policies: “A Global Cache should support a minimum of 100 GB of data in the cache”

https://wmo-im.github.io/wis2-guide/guide/wis2-guide-DRAFT.html#_procedure_for_registration_of_a_new_global_service

7.3. Global Discovery Catalogue Service testing

7.3.1. Preparations for the test

- backup metadata repository
- metrics can be backed up, but as they are also rebuilt after a service restart, no backup is necessary here
- the GDC broker shall be made accessible to to the WIS2 test Global Broker, with access via the credentials `everyone-dev/everyone-dev` for the purpose of testing

7.3.2. Entering the test environment

The GDC implementation under test (IUT) is required to enter the test environment with the following state:

- empty metadata repository (0 records)
- empty / non-existent metadata archive zipfile

7.3.3. Test setup

The GDC test setup consists of the following:

- A test data bundle to consist of:
  - 6 valid WCMP2 records
  - 4 broken JSON WCMP2 records
  - 1 invalid WCMP2 record
  - 20 WNM documents, each of which pointing to the related WCMP2 records
  - 1 WNM document specifying a WCMP2 record deletion
- all WCMP2 records stored on an HTTP server (GitHub)
  - documents available at the following location: https://github.com/wmo-im/wis2-global-services-testing/tree/main/tests/global_discovery_catalogue/metadata
  - all WNM documents updated to point to the correct HTTP server for proper HTTP dereferencing (raw GitHub) (example: https://raw.githubusercontent.com/wmo-im/wis2-global-services-testing/refs/heads/main/global-services-testing/sections/testing/global-discovery-catalogue.adoc)
- updating environment variables in `tests/default.env` as follows:
  - `GDC_API`: URL to GDC API (including collection name)
  - `GDC_CENTRE_ID`: the centre identifier of the GDC implementation under test (IUT)
  - `GB_CENTRE_ID`: the centre identifier of the GB used by the GDC implementation under test

(IUT)

Note that you can store secret environments in `tests/secrets.env` and refer to them in `tests/default.env` accordingly.

Given the GDC performs various checks on the centre identifier as part of the incoming topic from the Global Broker as well as the the WCMP2 identrifier check, GDC tests require a fixed WIS2 Node centre id.

The fixed WIS2 Node centre used for GDC tests is `io-wis2dev-11-test`.

7.3.4. Functional tests

7.3.4.1. Global Broker connection and subscription

Purpose

A Global Discovery Catalogue must connect to a Global Broker using the MQTT protocol with TLS and username/password authentication (everyone/everyone) and subscribe to the following topic:

- `cache/a/wis2/+/metadata/#`

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication
2. Connect the MQTT client to the Global Broker
3. Once the connection is successful, attempt to subscribe to the following topics:
    - `cache/a/wis2/+/metadata/#`
4. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails
5. Close the connections to the broker after the test

On successful completion, the following metrics should be modified:

- `wmo_wis2_gdc_connected_flag` for the centre-id from which the Global Discovery Catalogue connected to should be set to 1 (one)

7.3.4.2. Notification and metadata processing (success)

Purpose

The Global Discovery Catalogue should be able to process valid WCMP2 metadata record of core data published by a WIS2 Node.

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication.

2. Connect the MQTT client to the Global Broker
3. Once the connection is successful, attempt to subscribe to the following topics:
    - `cache/a/wis2/+/metadata/#`
4. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails
5. On an incoming message:
    a. find the canonical link object in the `links` array
    b. from the matching link, issue a HTTP GET request against the matching `href` value
    c. parse the HTTP response:
        i. validate against the WCMP2 Executable Test Suite (ETS)
        ii. publish the WCMP2 record to the catalogue
        iii. publish an ETS validation report as a notification message to the GDC MQTT broker
    d. using the WCMP2 identifier (i.e. `$WCMP2_ID`), construct a path to the record on the GDC (`https://HOST/collections/wis2-discovery-metadata/$WCMP2_ID`)

On successful completion:

- the resulting WCMP2 record should be available on the GDC API and contain an MQTT link / channel (using `cache/a/wis2`) foreach Global Broker
- the following metrics should be modified:
    - `wmo_wis2_gdc_passed_total` for the centre-id from where the metadata was published from should be incremented by 1 (one)
    - `wmo_wis2_gdc_core_total` for the centre-id from where the metadata (core data policy) was published from should be incremented by 1 (one)
- a notification message should arrive from the Global Broker under `monitor/a/wis2/CENTRE_ID_global-discovery-catalogue/centre-id`)

7.3.4.3. Notification and metadata processing (failure; record not found)

Purpose

The Global Discovery Catalogue should be able to process failing (record not found) WCMP2 metadata published by a WIS2 Node.

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication
2. Connect the MQTT client to the Global Broker
3. Once the connection is successful, attempt to subscribe to the following topics:
    - `cache/a/wis2/+/metadata/#`
4. Check if the subscription is successful. If the subscription is successful, proceed. If the

subscription is not successful, the test fails

5. On an incoming message:
   a. find the canonical link object in the `links` array
   b. from the matching link, issue a HTTP GET request against the matching `href` value
   c. if the response is an HTTP status code of 404:
      i. publish an ETS error report as a notification message to the GDC MQTT broker

On successful completion:

- the following metrics should be modified:
  - `wmo_wis2_gdc_failed_total` for the centre-id from where the metadata was published from should be incremented by 1 (one)
- a notification message should arrive from the Global Broker under `monitor/a/wis2/CENTRE_ID_global-discovery-catalogue/centre-id`)

7.3.4.4. Notification and metadata processing (failure; malformed JSON or invalid WCMP2)

Purpose

The Global Discovery Catalogue should be able to process failing (malformed JSON) WCMP2 metadata published by a WIS2 Node.

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication
2. Connect the MQTT client to the Global Broker.
3. Once the connection is successful, attempt to subscribe to the following topics:
   - `cache/a/wis2/+/metadata/#`
4. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails
5. On an incoming message:
   a. find the canonical link object in the `links` array.
   b. from the matching link, issue a HTTP GET request against the matching `href` value.
   c. parse the HTTP response:
   d. if the JSON is malformed, or the WCMP2 is invalid:
      i. publish an ETS error report as a notification message to the GDC MQTT broker.

On successful completion:

- the following metrics should be modified:
  - `wmo_wis2_gdc_failed_total` for the centre-id from where the metadata was published from should be incremented by 1 (one).

- a notification message should arrive from the Global Broker under `monitor/a/wis2/CENTRE_ID_global-discovery-catalogue/centre-id`)

7.3.4.5. Metadata ingest centre-id mismatch

Purpose

A Global Discovery Catalogue should detect a mismatch between an incoming message topic's centre-id and the centre-id as part of a WCMP2 record identifier.

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication
2. Connect the MQTT client to the Global Broker
3. Once the connection is successful, attempt to subscribe to the following topics:
   - `cache/a/wis2/+/metadata/#`
4. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails
5. On an incoming message:
   a. capture the centre-id from the topic (4th token split on `/`)
   b. find the canonical link object in the `links` array
   c. from the matching link, issue a HTTP GET request against the matching `href` value
   d. parse the HTTP response:
   e. extract the centre-id from WCMP2 record identifier (`id` property, 3rd token split on `:`)
   f. in the WCMP2 record, if a MQTT link exists (`rel=items`, `channel` starts with `origin/a/wis2`), capture the centre-id from the topic (4th token split on `/`)
6. compare the following values to verify that they are identical:
   a. centre-id extracted from topic
   b. centre-id extracted from WCMP2 identifier
   c. centre-id extracted from MQTT link in WCMP2 record
7. publish an ETS error report as a notification message to the GDC MQTT broker

On successful completion, the following metrics should be modified:

- `wmo_wis2_gdc_failed_total` for the centre-id from where the metadata was published from should be incremented by 1 (one)
- a notification message should arrive from the Global Broker under `monitor/a/wis2/CENTRE_ID_global-discovery-catalogue/centre-id`)

7.3.4.6. Notification and metadata processing (record deletion)

Purpose

The Global Discovery Catalogue should be able to process valid WCMP2 metadata record deletion of core data published by a WIS2 Node.

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication
2. Connect the MQTT client to the Global Broker
3. Once the connection is successful, attempt to subscribe to the following topics:
    - `cache/a/wis2/+/metadata/#`
4. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails
5. On an incoming message:
    a. find the link object in the `links` array where `rel=deletion`.
    b. capture the `properties.metadata_id` value
    c. from the matching link, issue a HTTP GET request against the matching `href` value.
    d. parse the HTTP response:
        i. validate against the WCMP2 Executable Test Suite (ETS)
        ii. delete the WCMP2 record from the catalogue using the value from `properties.metadata_id` captured earlier in the test
        iii. publish a notification message to the GDC MQTT broker
    e. using the WCMP2 identifier (i.e. `$WCMP2_ID`), construct a path to the record on the GDC (`https://HOST/collections/wis2-discovery-metadata/$WCMP2_ID`)

On successful completion:

- the WCMP2 record should be removed from the GDC API
- the following metrics should be modified:
    - `wmo_wis2_gdc_passed_total` for the centre-id from where the metadata was published from should be decremented by 1 (one)
    - `wmo_wis2_gdc_core_total` for the centre-id from where the metadata (core data policy) was published from should be decremented by 1 (one)
- a notification message should arrive from the Global Broker under `monitor/a/wis2/CENTRE_ID_global-discovery-catalogue/centre-id`)

7.3.4.7. Notification and metadata processing (failure; record deletion message does not contain `properties.metadata_id`)

Purpose

The Global Discovery Catalogue should be able to detect a WNM error when `properties.metadata_id`

is missing from a WCMP2 deletion request.

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication.
2. Connect the MQTT client to the Global Broker
3. Once the connection is successful, attempt to subscribe to the following topics:
   - `cache/a/wis2/+/metadata/#`
4. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails
5. On an incoming message:
   a. find the link object in the `links` array where `rel=deletion`
   b. capture the missing `properties.metadata_id` value
   c. publish a notification message of the error to the GDC MQTT broker

On successful completion:

- a notification message should arrive from the Global Broker under `monitor/a/wis2/CENTRE_ID_global-discovery-catalogue/centre-id`)

7.3.4.8. WCMP2 metadata archive zipfile publication

Purpose

Validate that a GDC API publishes a metadata archive zipfile.

Note that this test should only be executed if the GDC IUT has the ability to generate the metadata zipfile during the testing window.

Steps

1. Construct a path to the GDC endpoint (`https://HOST/collections/wis2-discovery-metadata`).
2. Issue a HTTP GET request on the path
3. Parse the HTTP response
4. Check that the record includes a `links` array
5. In the `links` array, check that a metadata archive zipfile link is available (where a link object's `rel=archives` and `type=application/zip`)
6. In the matching link, issue a HTTP GET request on the associated `href` value
7. Unzip the content of the HTTP response

On successful completion:

- the resulting HTTP response should be zip encoded data, which, when unzipped, contains a directory of JSON files of WCMP2 metadata

7.3.4.9. WCMP2 cold start initialization from metadata archive zipfile

Purpose

Validate that a GDC initializes from a metadata archive zipfile.

Note that this test should only be executed if the GDC IUT has the ability to generate the metadata zipfile during the testing window.

Steps

1. Construct a path to an existing, functional GDC endpoint (`https://HOST/collections/wis2-discovery-metadata`)
2. Issue a HTTP GET request on the path
3. Parse the HTTP response
4. Check that the record includes a `links` array
5. In the `links` array, check that a metadata archive zipfile link is available (where a link object's `rel=archives` and `type=application/zip`)
6. In the matching link, issue a HTTP GET request on the associated `href` value
7. Unzip the content of the HTTP response
8. Foreach WCMP2 (JSON) record in the zipfile, validate and ingest into the new GDC
9. Construct a path to the GDC endpoint (`https://HOST/collections/wis2-discovery-metadata/items`)
10. Issue a HTTP GET request on the path
11. Parse the HTTP response
12. Count the number of items in the `numberMatched` property

On successful completion:

- the number of the features in the GDC should match the number of records in the metadata archive zipfile.

7.3.4.10. API functionality

Purpose

Validate that a GDC API performs as expected based on the OGC API - Records standard.

Steps

1. Construct a path to the GDC endpoint (`https://HOST/collections/wis2-discovery-metadata`).
2. Issue a HTTP GET request on the path
3. Parse the HTTP response
4. Check that the record includes a `links` array
5. In the `links` array, check that an items link is available (where a link object's `rel=items` and `type=application/geo+json`)

6. In the matching link, issue a HTTP GET request on the associated `href` value
7. Parse the HTTP response
8. Ensure that a `numberMatched` property exists with an integer value of 6
9. Ensure that a `numberReturned` property exists with an integer value of 6
10. Construct a path to the GDC endpoint with a bounding box query parameter (`https://HOST/collections/wis2-discovery-metadata/items?bbox=-142,42,-53,84`)
11. Issue a HTTP GET request on the path
12. Parse the HTTP response
13. Ensure that a `numberMatched` property exists with an integer value of 2
14. Ensure that a `numberReturned` property exists with an integer value of 2
15. Ensure that a `features` array exists
16. Construct a path to the GDC endpoint with a temporal query parameter (`https://HOST/collections/wis2-discovery-metadata/items?datetime=2000-11-11T12:42:23Z/..`)
17. Issue a HTTP GET request on the path
18. Parse the HTTP response
19. Ensure that a `numberMatched` property exists with an integer value of 6
20. Ensure that a `numberReturned` property exists with an integer value of 6
21. Ensure that a `features` array exists
22. Construct a path to the GDC endpoint with a full text query parameter (`https://HOST/collections/wis2-discovery-metadata/items?q=observations`)
23. Issue a HTTP GET request on the path
24. Parse the HTTP response
25. Ensure that a `numberMatched` property exists with an integer value of 4
26. Ensure that a `numberReturned` property exists with an integer value of 4
27. Ensure that a `features` array exists

7.3.5. Performance tests

7.3.5.1. Processing timeliness

Purpose

Validate that a GDC is able to process WCMP2 metadata in a timely manner.

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication
2. Connect the MQTT client to the Global Broker
3. Once the connection is successful, attempt to subscribe to the following topics:

- `cache/a/wis2/+/metadata/#`

4. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails
5. On all incoming messages:
   a. find the canonical link object in the `links` array
   b. from the matching link, issue a HTTP GET request against the matching `href` value
   c. parse the HTTP response:
      i. validate against the WCMP2 Executable Test Suite (ETS)
      ii. publish the WCMP2 record to the catalogue
      iii. publish an ETS validation report as a notification message to the GDC MQTT broker.
   d. using the WCMP2 identifier (i.e. `$WCMP2_ID`), construct a path to the record on the GDC (`https://HOST/collections/wis2-discovery-metadata/$WCMP2_ID`)

On successful completion:

- all WCMP2 records should be processed and published in 5 minutes or less

7.3.6. Executing the test environment

To execute the core GDC functional tests:

```
# ensure that the Python virtual environment is activated
# NOTE: CENTRE_ID is the centre identifier of the IUT
export CENTRE_ID=ca-eccc-msc-global-discovery-catalogue
mkdir -p /data/wis2-testing/results/$CENTRE_ID
cd tests
pytest -s global_discovery_catalogue/test_gdc_functional.py -k "not zipfile"
--junitxml=/data/wis2-testing/results/$CENTRE_ID/$(date '+%Y-%m-%dT%H:%M:%SZ').xml -l
-rA | tee /data/wis2-testing/results/$CENTRE_ID/$(date '+%Y-%m-%dT%H:%M:%SZ').log
```

To execute the additional GDC functional tests:

```
# ensure that the Python virtual environment is activated which require interactive
steps from the GDC IUT.
cd tests
pytest -s global_discovery_catalogue/test_gdc_functional.py -k "zipfile"
--junitxml=/data/wis2-testing/results/$CENTRE_ID/$(date '+%Y-%m-%dT%H:%M:%SZ').xml -l
-rA | tee /data/wis2-testing/results/$CENTRE_ID/$(date '+%Y-%m-%dT%H:%M:%SZ').log  #
CENTRE_ID is the centre identifier of the IUT
```

To execute the GDC performance tests:

```
# ensure that the Python virtual environment is activated which require interactive
steps from the GDC IUT.
```

```
cd tests
pytest -s global_discovery_catalogue/test_gdc_performance.py --junitxml=/data/wis2
-testing/results/$CENTRE_ID/$(date '+%Y-%m-%dT%H:%M:%SZ').xml -l -rA | tee /data/wis2-
testing/results/$CENTRE_ID/$(date '+%Y-%m-%dT%H:%M:%SZ').log  # CENTRE_ID is the
centre identifier of the IUT
```

7.3.6.1. Useful flags

- `--sleep-factor`: some tests may require extra time for Pub/Sub workflow to complete. Passing the `--sleep-factor` (integer) option allows to set a multiplier applied to all sleep functions in the test.
- `-o log_cli=true log-level=DEBUG`: print logging messages to screen (useful for debugging).

7.3.7. Exiting the test environment

The GDC implementation under test (IUT) exits the test environment with the following state:

- restoration of resources at state prior to entry:
- metadata repository
- metrics endpoint
- metadata archive zipfile (create new archive file or restore from backed up file)

7.4. Global Monitor Service testing

7.4.1. Setup

A test data bundle to consist of:

- `1 valid endpoint with valid metrics.`
- `1 invalid endpoint.`
- `1 valid endpiont with malformed metrics.`
- `50 valid endpoints for the performance test.`

7.4.2. Functional tests

7.4.2.1. Connectivity

Purpose

A Global Monitor must connect to the endpoints provided by the other Global Services, and if any of the endpoints is unavailable, the error should be reported and the alert notification messages should arrive.

TODO

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication.
2. Connect the MQTT client to the broker of GM.
3. Once the connection is successful, attempt to subscribe to the following topics:
   - `monitor/a/wis2/#`
4. Initialize the GM with test endpoint with simulated metrics.
5. Check if the endpoint is available, if it is, it shows 'UP' on the webpage.
6. If the endpoint is unavailable, it shows error just like '404' on the webpage, and a notification message should arrive from the broker of GM under
   - `monitor/a/wis2/CENTRE_ID-global-monitor/centre-id`
     - e.g. monitor/a/wis2/cn-cma-global-monitor/us-noaa-nws-global-broker

7.4.2.2. Metrics validation failure; invalid

Purpose

The Global Monitor should be able to recognize the invalid metrics, give prompt and alert.

TODO

Steps

1. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication.
2. Connect the MQTT client to the broker of GM.
3. Once the connection is successful, attempt to subscribe to the following topics:
   - `monitor/a/wis2/#`
4. Initialize the GM with test endpoint with invalid metrics.
5. GM gives the error prompt on the webpage, and a notification message should arrive from the broker of GM under
   - `monitor/a/wis2/CENTRE_ID-global-monitor/centre-id`
     - e.g. monitor/a/wis2/ma-marocmeteo-global-monitor/us-noaa-nws-global-broker

7.4.2.3. Display the metrics on the dashboard

Purpose

The metrics that scraped from the Global Services could be displayed on the dashboard appropriately.

TODO

Steps

1. Initialize the GM with the endpoints with valid metrics.
2. Open the corresponding dashboard, if the dashboard could display the metrics in an appropriate way, it is successful.

7.4.2.4. Panel 1

Metrics:test_wis2_gb_connected_flag

Description:The connection status from the broker to the center is always 1

Query Used

test_wis2_gb_connected_flag{centre_id=~"cn-cma|ca-eccc-msc|cm-meteocameroon|it-meteoam|ms-metservice", report_by="cn-cma-global-broker"}

Result:

[dashboard 1] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-1.png*

7.4.2.5. Panel 2

Metrics:test_wis2_gb_connected_flag

Description:The connection status from the broker to the center is always 0

Query Used

test_wis2_gb_connected_flag{centre_id=~"kn-metservice|zm-zmd|sg-mss|na-meteona|uy-inumet", report_by=~"cn-cma-global-broker-bak"}

Result:

[dashboard 2] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-2.png*

7.4.2.6. Panel 3

Metrics:test_wis2_gb_connected_flag

Description:The connection status from the broker to the center alternates between 0 and 1 every minute

Query Used

test_wis2_gb_connected_flag{centre_id="ai-metservice", report_by="cn-cma-global-broker"}

Result:

[dashboard 3] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-*

testing/images/dashboard-3.png

7.4.2.7. Panel 4

Metrics:test_wis2_gb_connected_flag

Description:The connection status from the broker to the center alternates between 0 and 1 every five minute

Query Used

test_wis2_gb_connected_flag{centre_id="ar-smn", report_by="cn-cma-global-broker"}

Result:

[dashboard 4] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-4.png*

Query Used

test_wis2_gb_connected_flag{centre_id="au-bom", report_by="cn-cma-global-broker"}

Result:

[dashboard 5] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-5.png*

7.4.2.8. Panel 5

Metrics:test_wis2_gb_connected_flag

Description:All status values for the connection from the broker to the center

Query Used

test_wis2_gb_connected_flag{report_by="cn-cma-global-broker", centre_id=~"$centre_id"}

Result:

[dashboard 6 0] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-6-0.png*

7.4.2.9. Panel 6

Metrics:test_wis2_gb_last_message_timestamp_seconds

Description:Time difference between the Timestamp of last message received from centre and current time

Query Used

sort_desc(time()-wmo_wis2_gb_last_message_timestamp_seconds{centre_id="$centre_id",report_by="cn-cma-global-broker"})

Result:

[dashboard 6] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-6.png*

=====Panel 7 Metrics:test_wis2_gb_messages_received_total

Description: Total number of messages received by all center_id which report_by = cn-cma-global-broker

Query Used

sum by(report_by) (test_wis2_gb_messages_received_total{centre_id="$centre_id",report_by="$report_id"})

Result:

[dashboard 7] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-7.png*

7.4.2.10. Panel 8

Metrics:test_wis2_gb_messages_received_total

Description:Number of messages received by each center_id

Query Used

sum by(report_by) (test_wis2_gb_messages_received_total{centre_id="$centre_id",report_by="$report_id"})

Result:

[dashboard 8] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-8.png*

=====Panel 9 Metrics:test_wis2_gb_messages_invalid_topic_total

Description: Total number of invalid topic messages from all center_id which report_by = cn-cma-global-broker

Query Used

sum by(report_by) (test_wis2_gb_messages_invalid_topic_total{centre_id="$centre_id",report_by="$report_id"})

Result:

[dashboard 9] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-9.png*

7.4.2.11. Panel 10

Metrics:test_wis2_gb_messages_invalid_format_total

Description: Total number of invalid topic messages from all center_id which report_by = cn-cma-global-broker

Query Used

sum by(report_by) (test_wis2_gb_messages_invalid_format_total{centre_id="$centre_id",report_by="$report_id"})

Result:

[dashboard 10] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-10.png*

7.4.2.12. Panel 11

Metrics:test_wis2_gb_messages_no_metadata_total

Description: Total number of received without corresponding metadata from all center_id which report_by = cn-cma-global-broker

Query Used

sum by(report_by) (test_wis2_gb_messages_no_metadata_total{centre_id="$centre_id",report_by="$report_id"})

Result:

[dashboard 11] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-11.png*

7.4.2.13. Panel 12

Metrics:test_wis2_gb_messages_published_total

Description: Number of messages published by cn-cma-global-broker

Query Used

sum by(report_by) (test_wis2_gb_messages_published_total{centre_id="$centre_id",report_by="$report_id"})

Result:

[dashboard 12] | *https://github.com/wmo-im/wis2-global-services-testing/blob/main/global-services-testing/images/dashboard-12.png*

7.4.2.14. Raising alert 1

Purpose

The Global Monitor could raise the alert according to the metrics and the alerting rules.

TODO

Steps

1. Simulate the metrics, and set wmo_wis2_gb_connected_flag{centre_id="int-ecmwf"} = 0 reported by 3 Global Brokers.
2. Publish the metrics once per minute.
3. Configure gb.yml

alert: disconnectedwis2nodemultiplegb

expr: count by (centre_id) (wmo_wis2_gb_connected_flag == 0) > 1

for: 2m

labels:

severity: error

annotations:

summary: Disconnected WIS2 Node from multiple Global Brokers

1. Wait for 2 minutes, and watch the webpage, if the alert is raised on the webpage, it is successful, otherwise, it is unsuccessful.
2. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication.
3. Connect the MQTT client to the broker of GM.
4. Once the connection is successful, attempt to subscribe to the following topics:
   - `monitor/a/wis2/#`
5. An alert notification message should arrive from the broker of GM under
   - `++monitor/a/wis2/CENTRE_ID-global-monitor/int-ecmwf`

7.4.2.15. Raising alert 2

Purpose

The Global Monitor could raise the alert according to the metrics and the alerting rules.

TODO

Steps

1. Simulate the metrics, and set wmo_wis2_gc_downloaded_total = 0 reported by cn-cma-global-cache.
2. Publish the metrics once per minute.
3. Configure the gc.yml

No data is received by Global Cache over the two minutes

- alert: No_data

```
expr: sum by (report_by) (delta(wmo_wis2_gc_downloaded_total[2m])) == 0
```

```
for: 2m
```

```
labels:
```

```
severity: critical
```

```
annotations:
```

```
summary: The Global cache is not receiving any data since two minutes
```

1. Wait for 2 minutes, and watch the webpage, if the alert is raised on the webpage, it is successful, otherwise, it is unsuccessful.
2. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication.
3. Connect the MQTT client to the broker of GM.
4. Once the connection is successful, attempt to subscribe to the following topics:

- `monitor/a/wis2/#`
  1. An alert notification message should arrive from the broker of GM under
- `monitor/a/wis2/CENTRE_ID_global-monitor/cn-cma-global-cache`

7.4.3. Performance tests

7.4.3.1. Multiple providers

Purpose

A Global Monitor should support a minimum of 50 metrics providers.

TODO

Steps

1. Set up the configuration with 50 simulated endpoints.
2. If all the endpoints shows 'UP' on the webpage, the test passes.
3. Open the dashboard and check if it matches the metrics, if it is, the test passes.

7.4.3.2. Simultaneous access

Purpose

A Global Monitor should support 200 simultaneous access to the dashboard

TODO

Steps

1. Open Jmeter and configure the Test Plan:
    - GM address, username and password
    - threads(=200)
    - Ramp-up Time and Loop Count(=30s)
    - Add listener
    - Run the test
2. When the test finishes, look at the results in the listeners. Look at things like response time, throughput, and error rate.

7.5. GTS-to-WIS2 Gateway testing

No formal (e.g., automated) testing is required for the GTS-to-WIS2 Gateway.

Each WIS Centre seeking to decommission its GTS Message Switch must check that all the bulletins they normally receive via GTS are available from at least one GTS-to-WIS2 Gateway.

Each GISC should work with RTHs to validate that everything their affiliated centres are producing on GTS are available from each Gateway.

> The assessment of the GTS bulletins being pubished via the GTS-to-WIS2 Gateways should be completed between 2025-01-01 2025-03-31.

In summary, we propose to use “user” feedback to determine veracity and completeness of data published by each Gateway.

Performance testing is not required because each GTS-to-WIS2 Gateway has already demonstrated sustained throughput of bulletins to messages and data from GTS into WIS2.

7.6. WIS2-to-GTS Gateway testing

Each WIS2-to-GTS gateway instance will be deployed into the wis2dev.io environment and tested using automated scripts – following the approach used to test the main Global Services.

The WIS2-to-GTS gateway is functionally similar to the Global Cache – albeit with that it publishes to GTS rather than re-publishing cached data into WIS2. The Global Cache tests provide a good basis for testing the gateway.

There are 8 functional tests:

1. WIS2 Notification Message (WNM) Processing
2. Source Download Failure
3. Data Integrity Check Failure
4. WIS2 Notification Message Deduplication
5. WIS2 Notification Message Deduplication (Alternative 1)
6. WIS2 Notification Message Deduplication (Alternative 2)
7. Data Update
8. GTS Properties Validation Failure

Functional tests 1-7 are derived from similar tests developed for assessing the functional performance of a Global Cache.

These functional tests do not evaluate functionality of Message Switch as these components are already known to function correctly in operations. Furthermore, it may be difficult to deploy a functioning GTS Message Switch into the wis2dev.io environment.

WIS2-to-GTS Gatweway performance is assessed using one test:

1. WIS2 Notification Message Processing Rate

The performance test is based on “normal data processing” (see functional test “WIS2 Notification Message (WNM) Processing) with a throughput of 2000 unique WNMs per second that include valid `properties.gts` corresponding to the GTS Headers that the Gateway has whitelisted and trigger a download of data for onward distribution to the GTS.

This performance test is derived from a similar test developed for assessing the performance of a Global Cache.

Further WIS2-to-GTS gateway testing will be conducted during the “dry-run” MSS Decommissioning activity. Here we will also validate that the data from WIS2 is correctly packaged into bulletins by the gateway and propagated via the GTS.

> Note that the metrics referenced in these tests are proposed here, currently pending review: https://github.com/wmo-im/wis2-metric-hierarchy/issues/18

7.6.1. Preparation

7.6.1.1. Setup

Before running the pytest tests, ensure the following setup steps are completed:

Disconnect the Gateway from the production environment.

Ensure that the Gateway is not connected to the production environment during the testing session.

This ensures that all testing is completed in a controlled environment.

1. Remove other subscriptions to prod GB's.
    - Ensure that the Gateway is not subscribed to any other Global Broker except the dev/test Global Broker.

Connect the Gateway to the dev/test environment.

Ensure that the Gateway is connected to the dev/test environment during the testing session. This is to allow the test messages to be propagated to the Gateway for testing.

1. Gateway Subscription to the dev/test Global Broker.
    - Ensure the Gateway is subscribed to the dev/test Global Broker (`gb.wis2dev.io`).
        - the connection string to be used is: `mqtts://everyone-dev:everyone-dev@gb.wis2dev.io:8883`
    - The Gateway should be subscribed to the following topics:
        - `origin/a/wis2/+/data/#`
        - `cache/a/wis2/+/data/#`

Pytest Setup

Prior to executing the tests, update the `wis2-to-gts-gateway.env` file with the necessary environment variables. This env file is nested in the `tests/wis2-to-gts-gateway` directory and the values of the following variable must be updated:

- `WG_METRICS_REPORT_BY` - this is the centre_id of the Global Service and matches the report_by property in the metrics.
- `WG_DATA_OBJECTS_FOR_GTS_DISSEMINATION_LOCATIONS` - this is the list of directories where the Gateway will put data objects that have been downloaded ready for onward distribution via the GTS.

> TODO: The env file and other resources in the `tests/wis2-to-gts-gateway` directory need to be created.

7.6.1.2. Teardown

To teardown the configuration after the testing session, simply reverse the setup steps:

- Unsubscribe the Gateway from the dev/test Global Broker.
- Reset the metrics so that dev and prod metrics are not mixed.
- Reconnect the Gateway Cache to the production environment by re-subscribing to the production Global Brokers.

7.6.2. Functional Tests

7.6.2.1. WIS2 Notification Message (WNM) Processing

Purpose

Test that the Gateway functions as expected under normal conditions. The Gateway must process valid incoming WNMs that include `properties.gts` (TTAAii, CCCC) that correspond with the set of GTS Headers that are whitelisted, download the data at the provided canonical link, and pass the downloaded data object to a target location (directory) for onward publishing to the GTS by a Message Switch.

The Gateway must update the necessary metrics.

Note: This test does not evaluate functionality of Message Switch as these components are already known to function correctly in operations. Furthermore, it may be difficult to deploy a functioning GTS Message Switch into the wis2dev.io environment.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on `origin/a/wis2` and `cache/a/wis2` topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#
  - `cache/a/wis2/+/data/#
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and associated data objects:
  - A known number of valid WNM's with:
    - `properties.data_id` + `properties.pubtime should be unique to each message. Ensuring a different `data_id` is best here
    - `properties.integrity` with valid `properties.integrity.value` for the associated data object and a valid `properties.integrity.method`
    - Varying provision of `properties.gts`:
      - Some messages not including `properties.gts`
      - Some messages including `properties.gts` with valid GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
      - Some messages including `properties.gts` with valid GTS Headers (TTAAii, CCCC) that

are not whitelisted in the Gateway configuration

- Accompanying data objects should be accessible via the canonical link provided in the WNM.
  - The canonical link should be accessible per the core requirements

Steps

1. Configure the MQTT test client to connect to the dev/test GB MQTT broker using the provided connection string.
2. Publish a batch of Prepared WIS2 Notification Messages to the dev/test GB on following topics:
   - Send 1 or more messages to origin/a/wis2/+/data/#
   - Send 1 or more messages to cache/a/wis2/+/data/#
3. Assess the data objects downloaded by the Gateway and the Gateway Metrics

Evaluate

- Data Objects
  - The total number of data objects downloaded by the Gateway. This should match the number of notification messages published that include whitelisted GTS Headers (TTAAii, CCCC).
  - The data objects downloaded by the Gateway should be identical to the source data objects.
  - The diff or hashes of the data objects should be identical.
- Gateway Metrics
  - `wmo_wis2_wg_downloaded_total` (matches total number of messages that include whitelisted GTS Headers)
  - `wmo_wis2_wg_messages_gtsproperties_total` (matches total number of messages that include any valid GTS Headers)
  - `wmo_wis2_wg_messages_total` (matches total number of messages)
  - `wmo_wis2_wg_downloaded_errors_total` (no change)
  - `wmo_wis2_wg_integrity_failed_total` (no change)
  - `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (no change)
  - `wmo_wis2_wg_dataserver_status_flag` (set to 1 for each)
  - `wmo_wis2_wg_dataserver_last_download_timestamp_seconds` (set for each and within expected time range)

7.6.2.2. Source Download Failure

Purpose

Where a Gateway receives a valid WNM but is unable to download a data item from the location specified in a notification message (i.e., the source data server), the metric `wmo_wis2_wg_dataserver_status_flag` for the source data server should be set to 0 (zero).

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on `origin/a/wis2` and `cache/a/wis2` topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number of valid WNM's with:
    - invalid data download links (a link object's `href` property where `rel=canonical`)
    - `properties.data_id` + `properties.pubtime` should be unique to each message. Ensuring a different `data_id` is best here
    - Valid `properties.gts` with GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
  - Accompanying data objects are not required for this test.

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WNM's to the dev/test GB on one or more of the following topics:
   - Send 1 or more messages to `origin/a/wis2/+/data/#`
   - Send 1 or more messages to `cache/a/wis2/+/data/#`
3. Assess the data objects downloaded by the Gateway (zero) and the Gateway Metrics

Evaluate

- Data Objects
  - No data objects should be downloaded by the Gateway.
- Gateway Metrics
  - `wmo_wis2_wg_downloaded_total` (unchanged)
  - `wmo_wis2_wg_messages_gtsproperties_total` (matches total number of messages that include any valid GTS Headers)

- `wmo_wis2_wg_messages_total` (matches total number of messages)
- `wmo_wis2_wg_downloaded_errors_total` (+=1 for each WNM)
- `wmo_wis2_wg_integrity_failed_total` (no change)
- `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (no change)
- `wmo_wis2_wg_dataserver_status_flag` (set to 0 for each)
- `wmo_wis2_wg_dataserver_last_download_timestamp_seconds` (unchanged)

7.6.2.3. Data Integrity Check Failure

Purpose

A Gateway should validate the integrity of the resources it downloads and only accept data which matches the integrity value from the WIS Notification Message. If the WIS Notification Message does not contain an integrity value, a Gateway should accept the data as valid.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on origin/a/wis2 and cache/a/wis2 topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number of valid WMN's with:
    - `properties.data_id` + `properties.pubtime` should be unique to each message. Ensuring a different `data_id` is best here
    - `properties.integrity` with invalid `properties.integrity.value` for the associated data object and/or an invalid `properties.integrity.method`
    - Valid `properties.gts` with GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
  - Accompanying data objects should be accessible via the canonical link provided in the WNM.
    - The canonical link should be accessible per the core requirements

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WMN's to the dev/test GB on one or more of the following topics:
   - Send 1 or more messages to `origin/a/wis2/+/data/#`
   - Send 1 or more messages to `cache/a/wis2/+/data/#`
3. Assess the data objects downloaded by the Gateway (zero) and the Gateway Metrics

Evaluate

- Data Objects
  - No data objects should be downloaded by the Gateway.
- Gateway Metrics
  - `wmo_wis2_wg_downloaded_total` (unchanged)
  - `wmo_wis2_wg_messages_gtsproperties_total` (matches total number of messages that include any valid GTS Headers)
  - `wmo_wis2_wg_messages_total` (matches total number of messages)
  - `wmo_wis2_wg_downloaded_errors_total` (+=1 for each WNM)
  - `wmo_wis2_wg_integrity_failed_total` (+=1 for each WNM)
  - `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (no change)
  - `wmo_wis2_wg_dataserver_status_flag` (set to 1 for each where a data object was successfully downloaded before failing the integrity check)
  - `wmo_wis2_wg_dataserver_last_download_timestamp_seconds` (unchanged)

7.6.2.4. WIS2 Notification Message Deduplication

Purpose

A Gateway must ensure that only one instance of a notification message with a given unique identifier (id) is successfully processed.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on `origin/a/wis2` and `cache/a/wis2` topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`

- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number of valid WMN's with:
    - `properties.data_id` + `properties.pubtime` are NOT unique to each message, but shared by 2 or more messages
    - `properties.integrity` with valid `properties.integrity.value` for the associated data object and a valid `properties.integrity.method`
    - Valid `properties.gts` with GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
  - Accompanying data objects should be accessible via the canonical link provided in the WNM.
    - The canonical link should be accessible per the core requirements

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WMN's to the dev/test GB on one or more of the following topics:
   - Send 1 or more messages to `origin/a/wis2/+/data/#`
   - Send 1 or more messages to `cache/a/wis2/+/data/#`
3. Assess the data objects downloaded by the Gateway and the Gateway Metrics

Evaluate

- Data Objects
  - Only one data object should be downloaded per unique identifier which is defined as `properties.data_id` + `properties.pubtime`.
  - The data objects downloaded by the Gateway should be identical to the source data objects.
  - The diff or hashes of the data objects should be identical.
- Gateway Metrics
  - `wmo_wis2_wg_downloaded_total` (+1= for each unique identifier)
  - `wmo_wis2_wg_messages_gtsproperties_total` (matches total number of messages)
  - `wmo_wis2_wg_messages_total` (matches total number of messages)
  - `wmo_wis2_wg_downloaded_errors_total` (no change)
  - `wmo_wis2_wg_integrity_failed_total` (no change)
  - `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (no change)

- `wmo_wis2_wg_dataserver_status_flag` (set to 1 for each)
- `wmo_wis2_wg_dataserver_last_download_timestamp_seconds` (set for each and within expected time range)

> Question: Are `wmo_wis2_wg_messages_gtsproperties_total` and `wmo_wis2_wg_messages_total` the total number of messages, or the total number of unique messages (based on the "id" property of the message)?

7.6.2.5. WIS2 Notification Message Deduplication (Alternative 1)

Purpose

Where a Gateway fails to process a notification message relating to a given unique data object (`properties.data_id` + `properties.pubtime`), a Gateway should successfully process a valid, subsequently received notification message with the same unique data identifier.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on origin/a/wis2 and cache/a/wis2 topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number of valid WMN's with:
    - `properties.data_id` + `properties.pubtime` are NOT unique to each message, but shared by 2 or more messages
    - `properties.integrity` with valid `properties.integrity.value` for the associated data object and a valid `properties.integrity.method`
    - Valid `properties.gts` with GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
    - This defines a unique identifier message set.
    - For each unique identifier message set, the first published message should be invalid, or the data object inaccessible, and the second message/data object should be valid.

    - At least some of the accompanying data objects should be accessible via the canonical link provided in the WNM.
        - The canonical link should be accessible per the core requirements

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WMN's to the dev/test GB such that the invalid WNM for each unique data identifier is published first. One or more of the following topics can be used:
    - Send 1 or more messages to `origin/a/wis2/+/data/#`
    - Send 1 or more messages to `cache/a/wis2/+/data/#`
3. Assess the data objects downloaded by the Gateway and the Gateway Metrics

Evaluate

- Data Objects
    - Only one data object should be downloaded per unique identifier which is defined as `properties.data_id` + `properties.pubtime`.
    - The data objects downloaded by the Gateway should be identical to the source data objects.
    - The diff or hashes of the data objects should be identical.
- Gateway Metrics
    - `wmo_wis2_wg_downloaded_total` (+1= for each unique identifier)
    - `wmo_wis2_wg_messages_gtsproperties_total` (matches total number of messages)
    - `wmo_wis2_wg_messages_total` (matches total number of messages)
    - `wmo_wis2_wg_downloaded_errors_total` (+1= for each unique identifier)
    - `wmo_wis2_wg_integrity_failed_total` (no change)
    - `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (no change)
    - `wmo_wis2_wg_dataserver_status_flag` (set to 1 for each unique identifier)
    - `wmo_wis2_wg_dataserver_last_download_timestamp_seconds` (set for each and within expected time range)

7.6.2.6. WIS2 Notification Message Deduplication (Alternative 2)

Purpose

Related to the two previous tests, a Gateway should not process and downloaded a data item if it has already processed and downloaded a data item with the same `properties.data_id` and a `properties.pubtime` that is equal to or less than the `properties.pubtime` of the new data item. This test is an extension of the previous tests and can be conducted in conjunction with them.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on `origin/a/wis2` and `cache/a/wis2` topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number of valid WMN's with:
    - `properties.data_id` + `properties.pubtime` are NOT unique to each message, but shared by 2 or more messages
    - `properties.integrity` with valid `properties.integrity.value` for the associated data object and a valid `properties.integrity.method`
    - Valid properties.gts with GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
    - This defines a unique identifier message set.
    - For each unique identifier message set, the first published message should be invalid, or the data object inaccessible, and the second message/data object should be valid.
  - At least some of the accompanying data objects should be accessible via the canonical link provided in the WNM.
    - The canonical link should be accessible per the core requirements

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WMN's to the dev/test GB such that for each unique identifier message set, the first published message has a pubtime that is greater than or equal to the subsequent message/s. One or more of the following topics can be used:
   - Send 1 or more messages to `origin/a/wis2/+/data/#`
   - Send 1 or more messages to `cache/a/wis2/+/data/#`
3. Assess the data objects downloaded by the Gateway and the Gateway Metrics

Evaluate

- Data Objects
  - For each message set with a shared data_id, each data object should be downloaded by the Gateway and assuming that the `properties.pubtime` has been correctly set (decreasing or equal) for each message sent in chronological order.
  - The data objects downloaded by the Gateway should be identical to the source data objects.
  - The diff or hashes of the data objects should be identical.
- Gateway Metrics
  - `wmo_wis2_wg_downloaded_total` (+1= for each unique identifier)
  - `wmo_wis2_wg_messages_gtsproperties_total` (matches total number of messages)
  - `wmo_wis2_wg_messages_total` (matches total number of messages)
  - `wmo_wis2_wg_downloaded_errors_total` (+1= for each unique identifier)
  - `wmo_wis2_wg_integrity_failed_total` (no change)
  - `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (no change)
  - `wmo_wis2_wg_dataserver_status_flag` (set to 1 for each unique identifier)
  - `wmo_wis2_wg_dataserver_last_download_timestamp_seconds` (set for each and within expected time range)

7.6.2.7. Data Update

Purpose

A Gateway should treat notification messages with the same data item identifier (`properties.data_id`), but different publication times (`properties.pubtime`) as unique data items. A Gateway only download data objects with the same data item identifier when messages are sent in chronological order AND they are marked as updates. Data items with the same `properties.data_id` but a greater/later publication time AND a update link (`links['rel']='update'`), should be processed. Data items with the same `properties.data_id` but earlier or identical publication times should be ignored.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on `origin/a/wis2` and `cache/a/wis2` topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`
- MQTT test client

    - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and data objects
    - A known number of valid WMN's with:
        - `properties.data_id` + `properties.pubtime` are unique to each message, but the `properties.data_id` is shared by 2 or more messages and the pubtimes are different
        - For a given shared `data_id`, the message with the latest pubtime has a link object with `rel=update`
        - `properties.integrity` with valid `properties.integrity.value` for the associated data object and a valid `properties.integrity.method`
        - Valid `properties.gts` with GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
        - This defines a unique identifier message set
    - Accompanying data objects should be accessible via the canonical link provided in the WNM.
        - The canonical link should be accessible per the core requirements

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WMN's to the dev/test GB such for each unique identifier message set, the first published message has a pubtime that is less than the subsequent message/s and subsequent messages have a valid update link. One or more of the following topics can be used:
    - Send 1 or more messages to `origin/a/wis2/+/data/#`
    - Send 1 or more messages to `cache/a/wis2/+/data/#`
3. Assess the data objects downloaded by the Gateway and the Gateway Metrics

Evaluate

- Data Objects
    - For each message set with a shared `data_id`, each data object should be downloaded by the Gateway and assuming that the properties.pubtime as been correctly set (increasing) for each message sent in chronological order.
    - The data objects downloaded by the Gateway should be identical to the source data objects.
    - The diff or hashes of the data objects should be identical.
- Gateway Metrics
    - wmo_wis2_wg_downloaded_total (+1= for each unique identifier arriving in chronological order)
    - wmo_wis2_wg_messages_gtsproperties_total (matches total number of messages)

- wmo_wis2_wg_messages_total (matches total number of messages)
- wmo_wis2_wg_downloaded_errors_total (no change)
- wmo_wis2_wg_integrity_failed_total (no change)
- wmo_wis2_wg_messages_gtsproperties_invalid_format_total (no change)
- wmo_wis2_wg_dataserver_status_flag (set to 1)
- wmo_wis2_wg_dataserver_last_download_timestamp_seconds (set for each and within expected time range)

7.6.2.8. GTS Properties Validation Failure

Purpose

A Gateway should only process messages that have valid `properties.gts` (TTAAii, CCCC) that correspond to the set of whitelisted GTS Headers. A Gateway must validate `properties.gts` to ensure that the TTAAii and CCCC are syntactically correct. Messages without `properties.gts` are ignored (see test "WIS2 Notification Message (WNM) Processing").

This test assesses that messages with invalid `properties.gts` are discarded.

A valid `properties.gts` shall contain:

- 1 (and only 1) subproperty: `ttaaii`
- 1 (and only 1) subproperty: `cccc`
- Any other subproperties are ignored
- `properties.gts.ttaaii` shall comprise a sequence of exactly 4 alphabetic characters following by 2 numeric characters (6 characters in total)
- `properties.gts.cccc` shall comprise a sequence of exactly 4 alphabetic characters

Note that:

- Alphabetic character case is ignored.
- TTAAii and CCCC values are not cross-referenced against WMO Volume C1 to determine if they are valid GTS bulletin headers.

Example GTS properties object:

```
"properties": {
    ...
    "gts": {
        "ttaaii": "ISMN01",
        "cccc": "EGRR"
    }
}
```

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on `origin/a/wis2` and `cache/a/wis2` topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`
- MQTT test client
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
- Gateway metrics scraper
- Prepared WIS2 Notification Messages and data objects
  - A known number of valid WMN's with:
    - `properties.data_id` + `properties.pubtime` should be unique to each message. Ensuring a different `data_id` is best here
    - `properties.gts` where `properties.gts.ttaaii` and/or `properties.gts.cccc` are invalid
  - Accompanying data objects are not required for this test

Steps

1. Configure the MQTT test client to connect to the dev/test MQTT broker using the provided connection string.
2. Publish the prepared WMN's to the dev/test GB on one or more of the following topics:
   - Send 1 or more messages to `origin/a/wis2/+/data/#`
   - Send 1 or more messages to `cache/a/wis2/+/data/#`
3. Assess the data objects downloaded by the Gateway (zero) and the Gateway Metrics

Evaluate

- Data Objects
  - No data objects should be downloaded by the Gateway.
- Gateway Metrics
  - `wmo_wis2_wg_downloaded_total` (unchanged)
  - `wmo_wis2_wg_messages_gtsproperties_total` (unchanged – the GTS properties are invalid)
  - `wmo_wis2_wg_messages_total` (matches total number of messages)
  - `wmo_wis2_wg_downloaded_errors_total` (unchanged)
  - `wmo_wis2_wg_integrity_failed_total` (unchanged)

- `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (+=1 for each WNM)
- `wmo_wis2_wg_dataserver_status_flag` (unchanged)
- `wmo_wis2_wg_dataserver_last_download_timestamp_seconds` (unchanged)

7.6.3. Performance Tests

7.6.3.1. WIS2 Notification Message Processing Rate

Purpose

A Gateway shall be able to successfully process, on average, 2000 unique WNM's per minute with an average message size of 85kb. The noted WNM's/minute rate can be used as a performance indicator for the Gateway being tested.

The Gateway must process valid incoming WNMs that include `properties.gts` (TTAAii, CCCC) that correspond with the set of GTS Headers that are whitelisted, download the data at the provided canonical link, and pass the downloaded data object to a target location (directory) for onward publishing to the GTS by a Message Switch.

The Gateway must update the necessary metrics.

Note: This test does not evaluate functionality of Message Switch as these components are already known to function correctly in operations. Furthermore, it may be difficult to deploy a functioning GTS Message Switch into the wis2dev.io environment.

Requirements

- Dev/test GB MQTT broker connection string
  - MQTT user is able to read and write messages on `origin/a/wis2` and `cache/a/wis2` topics.
- Dev/test Gateway is configured to process a known set of GTS Headers (TTAAii, CCCC)
  - To-do: defined set of whitelisted CCCC and TTAAii headers
  - For this test, the Gateway should place all downloaded files into the same location (directory).
- Dev/test Gateway is initiated and connected to the dev/test GB with subscriptions to the following topics:
  - `origin/a/wis2/+/data/#`
  - `cache/a/wis2/+/data/#`
- MQTT test clients
  - Client should connect to the dev/test GB MQTT broker using the provided connection string to control the input and monitor the output.
  - The test clients should be distributed among several locations.
  - Each client should send (or generate) valid WIS2 Notification Messages and make the accompanying data objects available for download at rates specified in this performance test.

    - The test clients should publish a known (or predictable) number of messages in a given time.
    - The test clients should keep sending / generating messages at the specified rate for multiple minutes.
- Gateway metrics scraper
- WIS2 Notification Messages and associated data objects (these may be prepared or generated as needed):
    - A known number of valid WNM's with:
        - `properties.data_id` + `properties.pubtime` should be unique to each message. Ensuring a different `data_id` is best here
        - `properties.integrity` with valid `properties.integrity.value` for the associated data object and a valid `properties.integrity.method`
        - `properties.gts` with valid GTS Headers (TTAAii, CCCC) that are whitelisted in the Gateway configuration
    - Accompanying data objects should be accessible via the canonical link provided in the WNM.
        - The canonical link should be accessible per the core requirements
        - The data object hash should match the hash provided in the WNM if integrity properties are provided
        - Average message size should be 85kb

> Question: "Average message size" is the phrase used in the GC performance test - should this be referring to the average size of the data object associated with each message?

Steps

1. Initialise the trigger client and publish the WNM dataset configuration to the dev/test GB on the topic `config/a/wis2/wg_performance_test`.
2. Every 60-seconds, run a script (or similar) to count the number of data objects (files) that have been successfully downloaded to the location configured in the Gateway.
3. Print the following metrics:
    a. Number of data objects downloaded for each minute of the test
    b. Average number of data objects downloaded per minute during the test - excluding the first and last minutes of the test
    c. Total number of data objects downloaded during the test
4. Assess the Gateway Metrics - there should not have been any errors during this test

> Note: Obviously there is some work needed to get this set up in the same way as for the Global Cache tests.

Note: Unlike the Global Cache performance tests, we can't use published WNMs to monitor the test progress. Instead, we have to look at the rate at which data objects are downloaded.

Evaluate

- Data objects
  - The total number of data objects downloaded by the Gateway should match the number of messages published by the test clients.
  - This test does not require comparison between source and downloaded data objects.
- Processing rate
  - The average processing rate should exceed 2000 messages per minute during the middle phase of the test (i.e., excluding results from the first and last minutes).
- Gateway Metrics
  - `wmo_wis2_wg_downloaded_total` (matches total number of messages and count of downloaded files)
  - `wmo_wis2_wg_messages_gtsproperties_total` (matches total number of messages)
  - `wmo_wis2_wg_messages_total` (matches total number of messages)
  - `wmo_wis2_wg_downloaded_errors_total` (no change)
  - `wmo_wis2_wg_integrity_failed_total` (no change)
  - `wmo_wis2_wg_messages_gtsproperties_invalid_format_total` (no change)
  - `wmo_wis2_wg_dataserver_status_flag` (set to 1 for each)

Chapter 8. Results

Pre-testing was performed in September 2024. Final tests were performed in October 2024. Foreach Global Service tested, participants included:

- Global Service operator contact point
- WMO Secretariat
- Testing support/expertise as defined per Global Service

The following tables provide the results of all tests (in alphabetical order, grouped by Global Service type).

8.1. Global Broker results

8.1.1. cn-cma-global-broker test results

Date: 2024-10-18

8.1.1.1. Functional tests

Table 1. cn-cma-global-broker functional test results

Test	Pass	Comments
Port	yes	
Certificate	yes	manual check approve that the certificate is correct
Origin and Cache Read-Access	yes	
Deny write access to *origin/a/wis2/ and cache/a/wis2/* for everyone/everyone credentials	yes	
cluster redundancy	yes	
Discarding of duplicate messages	yes	
[Publishing a message using the cn-cma-global-broker from a different WIS2 Node]	yes	
Publishing messages from a WIS2 Node using valid topics (compliant with WIS2 Topic Hierarchy)	yes	

Test	Pass	Comments
Publishing messages from a WIS2 Node using invalid topics (not compliant with WIS2 Topic Hierarchy)	yes	
Publishes messages from a WIS2 Node on a *valid* topic without corresponding metadata	yes	
Verifying the compliance of a WIS2 Notification message	yes	

8.1.1.2. Performance tests

Table 2. cn-cma-global-broker test results

Test	Pass	Comments
Minimum number of WIS2 Nodes	yes	
Minimum number of subscribers	yes	
Minimum number of messages per second	yes	the test script was not workikng correctly, so tests has been carried out manualy

8.1.2. fr-meteofrance-global-broker test results

Date: 2024-09-30

8.1.2.1. Functional tests

Table 3. fr-meteofrance-global-broker functional test results

Test	Pass	Comments
Port	yes	
Certificate	yes	
Origin and Cache Read-Access	yes	
Deny write access to *origin/a/wis2/ and cache/a/wis2/* for everyone/everyone credentials	yes	
cluster redundancy	yes	
Discarding of duplicate messages	yes	

Test	Pass	Comments
Publishing a message using the centre_id from a different WIS2 Node	yes	
Publishing messages from a WIS2 Node using valid topics (compliant with WIS2 Topic Hierarchy)	yes	
Publishing messages from a WIS2 Node using invalid topics (not compliant with WIS2 Topic Hierarchy)	yes	
Publishes messages from a WIS2 Node on a *valid* topic without corresponding metadata	yes	
Verifying the compliance of a WIS2 Notification message	yes	

8.1.2.2. Performance tests

Table 4. fr-meteofrance-global-broker performance test results

Test	Pass	Comments
Minimum number of WIS2 Nodes	yes	the test script was not ready, so the performance tests were carried out manually
Minimum number of subscribers	yes	
Minimum number of messages per second	yes	

8.1.2.3. Recommendation

- To avoid loss of connections a proposal to use everyone/everyone for the public and use specific credentials for partners
- QoS=1 for partners
- Qos=0 for public

8.1.3. br-inmet-global-broker test results

Date: 2024-10-03

8.1.3.1. Functional tests

Table 5. br-inmet-global-broker functional test results

Test	Pass	Comments
Port	yes	
Certificate	yes	
Origin and Cache Read-Access	yes	
Deny write access to *origin/a/wis2/ and cache/a/wis2/* for everyone/everyone credentials	yes	
cluster redundancy	yes	
Discarding of duplicate messages	yes	
[Publishing a message using the br-inmet-global-broker from a different WIS2 Node]	yes	
Publishing messages from a WIS2 Node using valid topics (compliant with WIS2 Topic Hierarchy)	yes	
Publishing messages from a WIS2 Node using invalid topics (not compliant with WIS2 Topic Hierarchy)	yes	
Publishes messages from a WIS2 Node on a *valid* topic without corresponding metadata	yes	
Verifying the compliance of a WIS2 Notification message	yes	

8.1.3.2. Performance tests

Table 6. br-inmet-global-broker test results

Test	Pass	Comments
Minimum number of WIS2 Nodes	yes	1
Minimum number of subscribers	yes	1
Minimum number of messages per second	yes	1

(1) The performance tests wer carried out in three scenarios as follows

Test	Scenario	Comments
Low Performance	concur:8,tmout:20,msg_count:2, msg_delay:500 concur:8,tmout:20,msg_count:4, msg_delay:500 concur:8,tmout:20,msg_count:6, msg_delay:500 concur:8,tmout:40,msg_count:8, msg_delay:500 concur:8,tmout:40,msg_count:1 0,msg_delay:500 concur:8,tmout:40,msg_count:1 2,msg_delay:500	Expected:3200, Received:3200 Expected:6400, Received:6400 Expected:9600, Received:9600 Expected:12800, Received:12800 Expected:16000, Received:16000 Expected:19200, Received:19200
Medium Performance	concur:4,tmout:20,msg_count:4, msg_delay:100 concur:8,tmout:20,msg_count:8, msg_delay:100 concur:16,tmout:40,msg_count: 16,msg_delay:100 concur:32,tmout:40,msg_count: 32,msg_delay:100	Expected:3200, Received:3200 Expected:12800, Received:12800 Expected:51200, Received:51200 Expected:204800, Received:204800
High Performance	concur:8,tmout:30,msg_count:8, msg_delay:50 concur:16,tmout:30,msg_count: 16,msg_delay:50 concur:32,tmout:60,msg_count: 32,msg_delay:50 concur:64,tmout:90,msg_count: 64,msg_delay:50	Expected:12800, Received:12800 Expected:51200, Received:51200 Expected:204800, Received:204800 Expected:204800, Received: lost of messages

8.1.3.3. Decision

- Analysis of the results to adjust the parameters
- Only the low tests are taken into account for acceptance purposes
- Medium and high tests are just for performance information

8.1.4. us-noaa-global-broker test results

Date: 2024-10-17

8.1.4.1. Functional tests

Table 7. us-noaa-global-broker functional test results

Test	Pass	Comments
Port	yes	
Certificate	yes	
Origin and Cache Read-Access	yes	
Deny write access to *origin/a/wis2/ and cache/a/wis2/* for everyone/everyone credentials	yes	
cluster redundancy	yes	
Discarding of duplicate messages	yes	
Publishing a message using the centre_id from a different WIS2 Node	yes	
Publishing messages from a WIS2 Node using valid topics (compliant with WIS2 Topic Hierarchy)	yes	
Publishing messages from a WIS2 Node using invalid topics (not compliant with WIS2 Topic Hierarchy)	yes	
Publishes messages from a WIS2 Node on a *valid* topic without corresponding metadata	yes	
Verifying the compliance of a WIS2 Notification message	yes	

8.1.4.2. Performance tests

Table 8. CENTRE_ID test results

Test	Pass	Comments
Minimum number of WIS2 Nodes	yes	
Minimum number of subscribers	yes	
Minimum number of messages per second	yes	

(1) The performance tests wer carried out in three scenarios as follows

Test	Scenario	Comments
Low Performance	concur:8,tmout:20,msg_count:2,msg_delay:500 concur:8,tmout:20,msg_count:4,msg_delay:500 concur:8,tmout:20,msg_count:6,msg_delay:500 concur:8,tmout:40,msg_count:8,msg_delay:500 concur:8,tmout:40,msg_count:10,msg_delay:500 concur:8,tmout:40,msg_count:12,msg_delay:500	Expected:3200, Received:3200 Expected:6400, Received:6400 Expected:9600, Received:9600 Expected:12800, Received:12800 Expected:16000, Received:16000 Expected:19200, Received:19200
Medium Performance	concur:4,tmout:20,msg_count:4,msg_delay:100 concur:8,tmout:20,msg_count:8,msg_delay:100 concur:16,tmout:40,msg_count:16,msg_delay:100 concur:32,tmout:40,msg_count:32,msg_delay:100	Expected:3200, Received:3200 Expected:12800, Received:12800 Expected:51200, Received:51200 Expected:**203300** , Received:204800
High Performance	concur:8,tmout:30,msg_count:8,msg_delay:50 concur:16,tmout:30,msg_count:16,msg_delay:50 concur:32,tmout:60,msg_count:32,msg_delay:50 concur:64,tmout:90,msg_count:64,msg_delay:50	not tested

8.1.4.3. Decision

- Analysis of the results to adjust the parameters
- Only the low tests are taken into account for acceptance purposes
- Medium and high tests are just for performance information
- A performance Test to be scheduled for all the GBs (November/December)

8.2. Global Cache results

8.2.1. data-metoffice-noaa-global-cache-results test results

Date: 2024-10-07

8.2.1.1. Functional tests

Table 9. data-metoffice-noaa-global-cache-results functional test results

Test	Pass	Comments
MQTT Broker Connectivity	yes	
GC MQTT Broker Subscription	yes	
WIS2 Notification Message (WNM) Processing	yes	
Cache False Directive	yes	
Source Download Failure	yes	
[Data Integrity Check Failure (Recommended)]	yes	
WIS2 Notification Message Deduplication	yes	
WIS2 Notification Message Deduplication (Alternative 1)	yes	
WIS2 Notification Message Deduplication (Alternative 2)	yes	
Data Update	yes	
[GC Metrics]	yes	The tests shows that the metrics comparison failed, but the manual verification of “wmo_wis2_gc_downloaded_total” via Prometheus shows that this test has also been successful

8.2.1.2. Performance tests

Table 10. data-metoffice-noaa-global-cache-results performance test results

Test	Pass	Comments
WIS2 Notification Processing Rate	yes	
Concurrent client downloads	yes	

8.2.2. de-dwd-global-cache test results

Date: 2024-10-10

8.2.2.1. Functional tests

Table 11. de-dwd-global-cache functional test results

Test	Pass	Comments
MQTT Broker Connectivity	yes	
GC MQTT Broker Subscription	yes	
WIS2 Notification Message (WNM) Processing	yes	
Cache False Directive	yes	
Source Download Failure	yes	
[Data Integrity Check Failure (Recommended)]	yes	
WIS2 Notification Message Deduplication	yes	
WIS2 Notification Message Deduplication (Alternative 1)	yes	
WIS2 Notification Message Deduplication (Alternative 2)	yes	
Data Update	yes	
[GC Metrics]	yes	The tests shows that the metrics comparison failed, but the manual verification of “wmo_wis2_gc_downloaded_total” confirmed that this test has also been successful

8.2.2.2. Performance tests

Table 12. de-dwd-global-cache performance test results

Test	Pass	Comments
WIS2 Notification Processing Rate	yes	
Concurrent client downloads	yes	

8.2.3. cn-cma-global-cache test results

Date: 2024-10-09

8.2.3.1. Functional tests

Table 13. cn-cma-global-cache functional test results

Test	Pass	Comments
MQTT Broker Connectivity	yes	
GC MQTT Broker Subscription	yes	

Test	Pass	Comments
WIS2 Notification Message (WNM) Processing	yes	
Cache False Directive	yes	
Source Download Failure	yes	
[Data Integrity Check Failure (Recommended)]	yes	
WIS2 Notification Message Deduplication	yes	
WIS2 Notification Message Deduplication (Alternative 1)	yes	
WIS2 Notification Message Deduplication (Alternative 2)	yes	
Data Update	yes	
[GC Metrics]	yes	A first random check was fine for wmo_wis2_gc_downloaded_total (centre_id="io-wis2dev-20-test") with value before tests 400, after functional tests still 400, and after performance tests 600.

8.2.3.2. Performance tests

Table 14. cn-cma-global-cache performance test results

Test	Pass	Comments
WIS2 Notification Processing Rate	yes	
Concurrent client downloads	yes	

8.2.4. sa-ncm-global-cache test results

Date: 2024-11-21

8.2.4.1. Functional tests

Table 15. sa-ncm-global-cache functional test results

Test	Pass	Comments
MQTT Broker Connectivity	yes	
GC MQTT Broker Subscription	yes	

Test	Pass	Comments
WIS2 Notification Message (WNM) Processing	yes	
Cache False Directive	yes	
Source Download Failure	yes	
[Cache Override (Optional)]	yes	
[Data Integrity Check Failure (Recommended)]	yes	
WIS2 Notification Message Deduplication	yes	
WIS2 Notification Message Deduplication (Alternative 1)	yes	
WIS2 Notification Message Deduplication (Alternative 2)	yes	
Data Update	yes	

8.2.4.2. Performance tests

Table 16. sa-ncm-global-cache performance test results

Test	Pass	Comments
WIS2 Notification Processing Rate	yes	
Concurrent client downloads	yes	

8.2.5. jp-jma-global-cache test results

Date: 2024-11-21

8.2.5.1. Functional tests

Table 17. jp-jma-global-cache functional test results

Test	Pass	Comments
MQTT Broker Connectivity	yes	
GC MQTT Broker Subscription	yes	
WIS2 Notification Message (WNM) Processing	yes	
Cache False Directive	yes	
Source Download Failure	yes	
[Cache Override (Optional)]	yes	

Test	Pass	Comments
[Data Integrity Check Failure (Recommended)]	yes	
WIS2 Notification Message Deduplication	yes	
WIS2 Notification Message Deduplication (Alternative 1)	yes	
WIS2 Notification Message Deduplication (Alternative 2)	yes	
Data Update	yes	

8.2.5.2. Performance tests

Table 18. jp-jma-global-cache performance test results

Test	Pass	Comments
WIS2 Notification Processing Rate	yes	
Concurrent client downloads	yes	the sleep factor has been increased but the test failed A manual test shows that downloading took a very long time

8.2.6. kr-kma-global-cache test results

Date: 2024-10-22

8.2.6.1. Functional tests

Table 19. kr-kma-global-cache functional test results

Test	Pass	Comments
MQTT Broker Connectivity	yes	
GC MQTT Broker Subscription	yes	
WIS2 Notification Message (WNM) Processing	yes	
Cache False Directive	yes	
Source Download Failure	yes	
[Cache Override (Optional)]	yes	
[Data Integrity Check Failure (Recommended)]	yes	

Test	Pass	Comments
WIS2 Notification Message Deduplication	yes	
WIS2 Notification Message Deduplication (Alternative 1)	yes	
WIS2 Notification Message Deduplication (Alternative 2)	yes	
Data Update	yes	

8.2.6.2. Performance tests

Table 20. kr-kma-global-cache performance test results

Test	Pass	Comments
WIS2 Notification Processing Rate	yes	
Concurrent client downloads	yes	

8.3. Global Discovery Catalogue results

8.3.1. ca-eccc-msc-global-discovery-catalogue test results

Date: 2024-10-23

8.3.1.1. Functional tests

Table 21. ca-eccc-msc-global-discovery-catalogue functional test results

Test	Pass	Comments
Global Broker connection and subscription	yes	
Notification and metadata processing (success)	yes	
Notification and metadata processing (failure; record not found)	yes	
Notification and metadata processing (failure; malformed JSON or invalid WCMP2)	yes	
Metadata ingest centre-id mismatch	yes	The sleep factor has been increased to 4 (from 3)
Notification and metadata processing (record deletion)	yes	The sleep factor has been increased to 4 (from 3)

Test	Pass	Comments
Notification and metadata processing (failure; record deletion message does not contain `properties.metadata_id`)	yes	The sleep factor has been increased to 4 (from 3)
WCMP2 metadata archive zipfile publication	yes	
WCMP2 cold start initialization from metadata archive zipfile	yes	
API functionality	yes	The sleep factor has been increased to 4 (from 3)

8.3.1.2. Performance tests

Table 22. ca-eccc-msc-global-discovery-catalogue performance test results

Test	Pass	Comments
Processing timeliness	yes	The sleep factor has been increased to 4 (from 3)

8.3.2. cn-cma-global-discovery-catalogue results

Date: 2024-10-21

8.3.2.1. Functional tests

Table 23. cn-cma-global-discovery-catalogue functional test results

Test	Pass	Comments
Global Broker connection and subscription	yes	
Notification and metadata processing (success)	yes	
Notification and metadata processing (failure; record not found)	yes	
Notification and metadata processing (failure; malformed JSON or invalid WCMP2)	yes	
Metadata ingest centre-id mismatch	yes	The sleep factor has been increased to 4 (from 3)
Notification and metadata processing (record deletion)	yes	The sleep factor has been increased to 4 (from 3)

Test	Pass	Comments
Notification and metadata processing (failure; record deletion message does not contain `properties.metadata_id`)	yes	The sleep factor has been increased to 4 (from 3)
WCMP2 metadata archive zipfile publication	yes	
WCMP2 cold start initialization from metadata archive zipfile	yes	
API functionality	yes	The sleep factor has been increased to 4 (from 3)

8.3.2.2. Performance tests

Table 24. cn-cma-global-discovery-catalogue performance test results

Test	Pass	Comments
Processing timeliness	yes	The sleep factor has been increased to 4 (from 3)

8.3.3. de-dwd-global-discovery-catalogue results

Date: 2024-10-23

8.3.3.1. Functional tests

Table 25. de-dwd-global-discovery-catalogue functional test results

Test	Pass	Comments
Global Broker connection and subscription	yes	
Notification and metadata processing (success)	yes	
Notification and metadata processing (failure; record not found)	yes	
Notification and metadata processing (failure; malformed JSON or invalid WCMP2)	yes	
Metadata ingest centre-id mismatch	yes	The sleep factor has been increased to 4 (from 3)
Notification and metadata processing (record deletion)	yes	The sleep factor has been increased to 4 (from 3)

Test	Pass	Comments
Notification and metadata processing (failure; record deletion message does not contain `properties.metadata_id`)	yes	The sleep factor has been increased to 4 (from 3)
WCMP2 metadata archive zipfile publication	yes	
WCMP2 cold start initialization from metadata archive zipfile	yes	
API functionality	yes	The sleep factor has been increased to 4 (from 3)

8.3.3.2. Performance tests

Table 26. de-dwd-global-discovery-catalogue performance test results

Test	Pass	Comments
Processing timeliness	yes	The sleep factor has been increased to 4 (from 3)

8.4. Global Monitor results

8.4.1. cn-cma-global-monitor test results

Date: 2024-10-24

8.4.1.1. Functional tests

Table 27. cn-cma-global-monitor functional test results

Test	Pass	Comments
Connectivity	yes	
[Metrics validation]	yes	
[Disply the metrics on the dashboard]	yes	
[Raising alert1(error): raise an alert by sending a message.]	yes	
[Raising alert2(critical): raise an alert by sending a message and issue a ticket to IMS.]	yes	

8.4.1.2. Performance tests

Table 28. cn-cma-global-monitor test results

Test	Pass	Comments
[Multiple providers: A Global Monitor should support a minimum of 50 metrics providers.]	yes	
[Simultaneous access: A Global Monitor should support 200 simultaneous access to the dashboard.]	yes	
[Bandwidth limitation: Could limit the bandwidth usage of the service to 100 Mb/s.]	yes	

8.4.2. ma-marocmeteo-global-monitor test results

Date: 2024-10-24

8.4.2.1. Functional tests

Table 29. ma-marocmeteo-global-monitor functional test results

Test	Pass	Comments
Connectivity	yes	
[Metrics validation]	yes	
[Disply the metrics on the dashboard]	yes	
[Raising alert1(error): raise an alert by sending a message.]	yes	
[Raising alert2(critical): raise an alert by publishing a message and sending a ticket to IMS.]	yes	

8.4.2.2. Performance tests

Table 30. ma-marocmeteo-global-monitor test results

Test	Pass	Comments
[Multiple providers: A Global Monitor should support a minimum of 50 metrics providers.]	yes	
[Simultaneous access: A Global Monitor should support 200 simultaneous access to the dashboard.]	yes	

Test	Pass	Comments
[Bandwidth limitation: Could limit the bandwidth usage of the service to 100 Mb/s.]	yes	

Chapter 9. Discussion

TODO

Chapter 10. Conclusions

TODO

Chapter 11. Future work

TODO

Appendix A: Revision History

Date	Release	Author	Primary clauses modified	Description
2024-03-30	0.1	Kralidis	all	initial version

Bibliography

- ab - Apache HTTP server benchmarking tool (2024) [1]

[1] https://httpd.apache.org/docs/2.4/programs/ab.html