WIS2 Global Services testing

World Meteorological Organization

Date: 2024-07-21

Version: 2024-03-28

Document location: https://community.wmo.int/wis2-global-services-testing

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

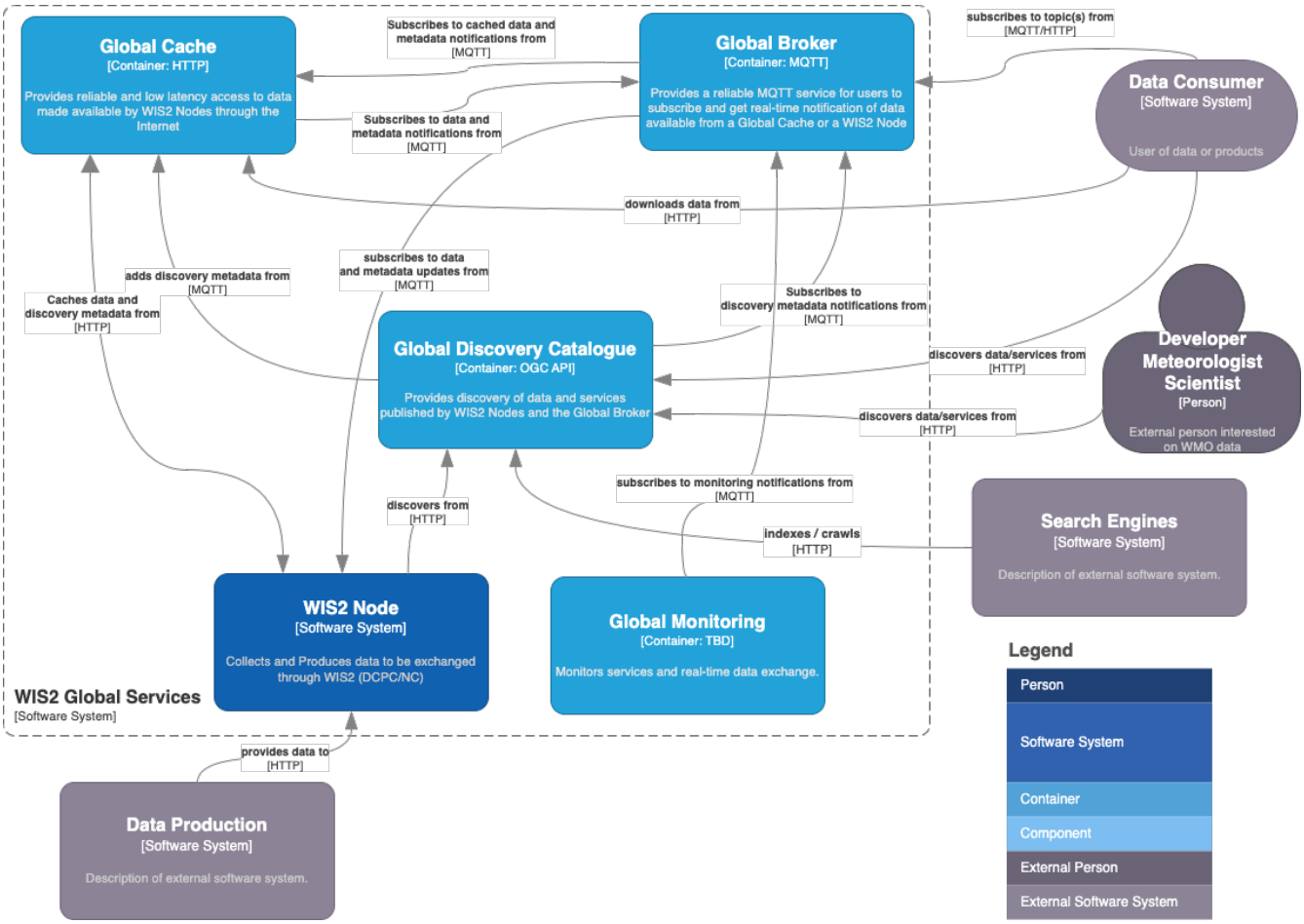

Figure 1. High Level Overview of the WIS2 Architecture

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

TODO

6.3.2. Environment

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

6.3.3. Performance testing

Ensure WIS2 Global Services are able to operate under various loads.

6.3.4. Functional testing

Ensure WIS2 Global Services operate with one another as expected and meet requirements.

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. Functional tests

7.1.1.1. 1. Global Broker Connectivity Tests

There are several check-box activities that WMO Secretariat can perform on each GB service in advance of the planned GB testing period (currently scheduled for September 30 - October 4).

A. Global Broker Port Tests

Purpose

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

Requirements

- Global Broker MQTT connection string
- MQTT Test Client

Steps

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

B. Global Broker Certificate Test

Purpose

The Global Broker service must use a valid HTTPS certificate. Transport Layer Security (TLS) is an encryption protocol that provides secure connections between servers and applications on the internet. When you connect to a website that uses HTTPS, TLS verifies the identity of the website's server and encrypts your connection, preventing hackers from intercepting any data transmitted between your device and the site. In this way, TLS provides privacy and security for the data transmitted between your browser and the website, and you can be sure the data hasn't been tampered with.

Requirements

- Global Broker MQTT connection string
- MQTT Test Client

Steps

1. From a MQTT client and try to connect to a GB using HTTPS. The GB server 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.

Evaluate

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

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.

C. Global Broker Origin and Cache Read-Access Test

Purpose

The Global Broker service must allow 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 for a 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

1. Check if the connection is successful, depending on the temporal frequency of updates, eventually, the WTH will be populated, and the test passes. If the connection is not successful, the test fails.

D. Global Broker deny write access to Origin and Cache for everyone/everyone credentials Test

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

1. 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.

E. Global Broker cluster redundancy Test

Purpose

The GB service, if 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 subcription 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.

7.1.1.2. 2. Global Broker Antiloop Testing

The antiloop feature of a Global Broker is a critical aspect for a successful pub/sub service. These 7 tests are designed to test the antiloop feature of the GB service. This must be fully functional for each WIS2 Global Broker properly prior WIS2 going to an operational state on January 1st, 2025

A. Discarding of duplicate messages Test

Purpose

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

Requirements

- Global Broker MQTT connection string to 2 WIS2 Nodes (test1 and test2)
- MQTT Test Client

Steps

1. Have WIS2Node "test1" publish **X** messages with the same id on topic **origin/a/wis2/test1/ core/data/weather/surface-based-observation/synop**
2. Have WIS2Node "test2" publish **Y** messages with the same id (same as id from X above) on topic **origin/a/wis2/test2/ core/data/weather/surface-based-observation/synop**

Evaluate

1. If the Global Broker "gb-test" discards all messages except one, makes it available on one of the two topics depending the WIS2 Node message that arrived first (**test1** or *test2)
2. Increments **wmo_wis2_gb_messages_published** on centre_id from the WIS2Node that arrives first (**test1** or **test2**)
3. If both statements are true, the test passes. Otherwise, the test fails.

B. Publishing a message using the centre_id from a different WIS2 Node Test

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

- Global Broker MQTT connection string to 2 WIS2 Nodes (test1 and test2)
- MQTT Test Client

Steps

1. Have WIS2Node "test1" publish a valid message on topic **origin/a/wis2/test1/core/data/weather/surface-based-observation/synop**
2. Have WIS2Node "test2" publish the same message and on topic **origin/a/wis2/test1/core/data/weather/surface-based-observation/synop**

Evaluate

1. If the Global Broker "gb-test" discards the message from **test2**, AND increments **wmo_wis2_gb_messages_invalid_total** on centre_id **test2**, the test passes. Otherwise, the test fails.

C. Publishing a message from a WIS2 Node using a compliant topic hierarchy Test

Purpose

The Global Broker service must check that the topic used to publish a message by a WIS2 Node is compliant with the agreed topic hierarchy. What follows below is an overview of the primary topic levels.

Level 1	channel
Level 2	version
Level 3	system
Level 4	centre-id
Level 5	notification-type
Level 6	data-policy
Level 7	earth-system-discipline

Requirements

- Global Broker MQTT connection string to 1 WIS2 Nodes (test1)
- MQTT Test Client

Steps

1. Have WIS2Node "test1" publish a **valid** message with valid topic hierarchy coming on topic **origin/a/wis2/test1/core/data/weather/surface-based-observation/synop**
2. Have WIS2Node "test1" then publish an **invalid message** with invalid topic hierarchy on topic **origin/wis2/test1/core/data/weather/surface-based-observation/synop**. What's missing from this topic hierarchy is **level 2** ("a").

Evaluate

1. If the Global Broker "gb-test" accepts the first message (with valid topic hierarchy) coming from WIS2 Node **test1**
2. If the Global Broker "gb-test" fails the second message coming from WIS2 Node **test1** with an

invalid topic hierarchy

3. If the Global Broker "gb-test" increments **wmo_wis2_gb_messages_invalid_total**, the test passes. Otherwise, the test fails.

D. Check that a GB subsribed to a WIS2 Node and publishes a message announces a dataset with metadata Test

Purpose

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

Requirements

- Global Broker MQTT connection string to 1 WIS2 Nodes (test1)
- MQTT Test Client

Steps

1. Using an MQTT client, subscribe to WIS2 Node "test1" on topic **origin/a/wis2/test1/core/data/weather/surface-based-observation/synop**
2. Have WIS2Node "test1" publish a valid message coming on topic **origin/a/wis2/test1/core/data/weather/surface-based-observation/synop**
3. Using the same MQTT client (like MQTT Explorer), check to see that that the WIS2 Node "test1" has updated it's available offerings. Check the message payload response to ensure this announcement

Evaluate

1. If the Global Broker "gb-test" sees the updated available offerings from the WIS2 Node "test1",
2. If the Global Broker verifies that the message payload response contains metadata and a link to the data, then this test passes. Otherwise, the test fails AND the GB will increment **wmo_wis2_gb_messages_no_metadata_total**.

E. Publishing and verifying the compliance of a WIS2 Notification message Test

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

- Global Broker MQTT connection string to 1 WIS2 Nodes (test1)
- MQTT Test Client

Steps

1. Have WIS2 Node "test1" publish a valid message for topic

origin/a/wis2/test1/core/data/weather/surface-based-observation/synop

2. Next have the same WIS2 Node "test1" publish an invalid WIS2 Notification Message on the same topic **origin/a/wis2/test1/core/data/weather/surface-based-observation/synop**

Evaluate

1. If the Global Broker "gb-test" accepts the first message published by WIS2 Node "test1" shows up as a valid WIS2 Notification Message
2. If the Global Broker "gb-test" drops the invalid WIS2 Notificaiton Message
3. If the Global Broker "gb-test" increments **wmo_wis2_gb_messages_invalid_total**, then this test passes. Otherwise, the test fails.

F. GB discards or notify in a notification message breaching the requirements laid out in tests C, D or E Tests

Purpose

The Global Broker service must be able to discard or notify via a notification message in breach of the requirement in tests C, D or E

Requirements

- Global Broker MQTT connection string to 1 WIS2 Node (test1)
- MQTT Test Client

Steps

1. Follow the steps outlined in tests C, D or E

Evaluate

1. If the steps from tests C or E are followed, and the test passes, the GB metrics should increment the **wmo_wis2_gb_messages_invalid_total** counter.
2. If the steps from test D are followed, and the test passes, the GB metrics should increment the **wmo_wis2_gb_messages_no_metadata_total** counter.

G. GB providing metrics Test

Purpose

The Global Broker service must provided metrics as requested in the Guide on WIS2

Requirements

- Global Broker MQTT connection string to 1 WIS2 Node (test1)
- MQTT Test Client

Steps

1. Have the GB update metrics for the first period of testing time (15 minutes) as usual

2. During the second period of time, make the metrics unavailable

Evaluate

1. If the GB provide metrics during the first period of testing time, AND doesn't provide metrics for the second period of testing time, the test passes. Otherwise, the test fails.

7.1.1.3. 3. Global Broker Performance Testing

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

A. Global Broker minimum number of WIS2 Nodes Test

Purpose

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

Requirements

- mqttx-cli
- hive
- global broker Cloud asset

Steps

1. Use hive/mqttx-cli scripting provided by Remy to perform this test

Evaluate

1. If the Global Broker subscribes to all 200 WIS2 Node subscriptions, the test passes. Otherwise, the test fails.

B. Global Broker minimum number of subscribers Test

Purpose

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

Requirements

- mqttx-cli
- hive
- global broker Cloud asset

Steps

1. Use hive/mqttx-cli scripting provided by Remy to perform this test

Evaluate

1. If the Global Broker supports a minimum of **1000** subscribers, the test passes. Otherwise, the test fails.

C. Global Broker minimum number of subscribers Test

Purpose

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

Requirements

- mqttx-cli
- hive
- global broker Cloud asset

Steps

1. Use hive/mqttx-cli scripting provided by Remy to perform this test

Evaluate

1. If the Global Broker supports a minimum of **10000** messages per second, the test passes. Otherwise, the test fails.

Chapter 8. Global Cache Service testing

8.1. Functional Tests

8.1.1. 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.

8.1.2. 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.

8.1.3. 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 GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly).
- Dev/test GB MQTT broker connection string
  - User is able to publish (write) messages to the dev/test GB on the `origin/a/wis2/` and `cache/a/wis2/` topic.
- 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 both the dev/test GB and GC MQTT brokers using the provided connection strings 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 and GC MQTT brokers using the provided connection strings.
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)

8.1.4. 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 GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly)
- Dev/test GB MQTT broker connection string
  - User is able to publish messages to the dev/test GB on the `origin/a/wis2/ and cache/a/wis2/` topic.
- 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 and GC MQTT brokers using the provided connection strings 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 and GC MQTT brokers using the provided connection strings.
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)

8.1.5. 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 GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly)
- Dev/test GB MQTT broker connection string
  - User is able to publish messages to the dev/test GB on the `origin/a/wis2/ and cache/a/wis2/` topic.
- 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 and GC MQTT brokers using the provided connection strings 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 GB and GC MQTT brokers using the provided connection strings.
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)

8.1.6. 6. Cache Override (Optional)

Purpose

Where a Global Cache determines that it is unable to cache a data item, 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, and the metric `wmo_wis2_gc_cache_override_total` is incremented by 1 (one). Note that the trigger for this directive is implementation specific. The criteria must be known and enabled for the test to be valid. Additionally, a given GC may decide to NOT implement this directive and thus this test is included as optional.

Requirements

- Dev/test GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly)
- Dev/test GB MQTT broker connection string
  - User is able to publish messages to the dev/test GB on the `origin/a/wis2/ and cache/a/wis2/` topic.
- 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 and GC MQTT brokers using the provided connection strings 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 true
    - `properties.data_id` + `properties.pubtime` should be unique to each message.
    - The known properties that trigger the cache override directive.
  - Accompanying data objects are not required for this test.

Steps

1. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings.
2. Publish the perpared 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

- Topic
  - No messages should be published on the `cache/a/wis2/#` topic as received by the test MQTT client.
- 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
  - The following metrics are updated as expected per the prepared test data set:
    - `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_cache_override_total` (+=1 for each WNM)
    - `wmo_wis2_gc_downloaded_errors_total` (unchanged)

8.1.7. 7. Data Integrity Check Failure (Recommended)

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 GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly)

- Dev/test GB MQTT broker connection string
  - User is able to publish messages to the dev/test GB on the `origin/a/wis2/ and cache/a/wis2/` topic.
- 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 and GC MQTT brokers using the provided connection strings 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. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings.
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)
  - `wmo_wis2_gc_integrity_failed_total` (+=1 for each WNM)

8.1.8. 8. 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 GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly)
- Dev/test GB MQTT broker connection string
  - User is able to publish messages to the dev/test GB on the `origin/a/wis2/ and cache/a/wis2/` topic.
- 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 and GC MQTT brokers using the provided connection strings 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. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings.
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

    - 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)

8.1.9. 8.1. 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 GC MQTT broker connection string
    - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly)
- Dev/test GB MQTT broker connection string
    - User is able to publish messages to the dev/test GB on the `origin/a/wis2/ and cache/a/wis2/` topic.
- 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 and GC MQTT brokers using the provided connection strings 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. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings.
2. 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)

8.1.10. 8.2. 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. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings.
2. 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)

8.1.11. 9. 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 GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly)
- Dev/test GB MQTT broker connection string
  - User is able to publish messages to the dev/test GB 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 and GC MQTT brokers using the provided connection strings 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. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings.
2. 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)

8.2. Performance tests

8.2.1. 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 75kb. This test represents the upper end of the current WNM volume. This test is a measured performance test similar to test 3. WNM Processing except that a large batch of messages is used, and the time taken to process the messages is measured. The noted WNM's/minute rate can be used as a performance indicator for the GC being tested.

Requirements

- Dev/test GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly).
- Dev/test GB MQTT broker connection string
  - User is able to publish (write) messages to the dev/test GB on the `origin/a/wis2/` and `cache/a/wis2/` topic.
- 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 both the dev/test GB and GC MQTT brokers using the provided connection strings 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. 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 75kb.

Steps

1. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings. This script and client will be used to orchestrate the test.
2. Start the timer, and publish the batch of 2000 prepared WNM's to the dev/test GB on following topics:
    - origin/a/wis2/+/data
    - cache/a/wis2/+/data
    - origin/a/wis2/+/metadata
    - cache/a/wis2/+/metadata
3. The test MQTT client should count the messages received on the `cache/a/wis2/#` topic that are published by the GC, but should not download the data objects.
4. Stop the timer when the MQTT client has received all expected messages (2000). A timeout can be set to allow the test to run as long as needed within a reasonable window.

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.

8.2.2. 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 GC MQTT broker connection string
  - User is able to connect to the GC MQTT broker and subscribe to the `cache/a/wis2/#` topic (readonly).
- Dev/test GB MQTT broker connection string
  - User is able to publish (write) messages to the dev/test GB on the `origin/a/wis2/` and `cache/a/wis2/` topic.
- 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 both the dev/test GB and GC MQTT brokers using the provided connection strings to control the input and monitor the output.
- Prepared WIS2 Notification Messages and associated data objects:
  - A known number (5) **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.
  - 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 500MB data object is recommended.

- Jmeter, Locust, or similar tool to manage the concurrent downloads.

Steps

1. Configure the MQTT test client to connect to the dev/test GB and GC MQTT brokers using the provided connection strings. This script and client will be used to orchestrate the test.
2. Publish the prepared WNM's one at a time 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

For each WNM:

1. Once the *cache* notification message is received by the test MQTT client (from the dev/test GC), the test client should start 1000 concurrent downloads of the data object/s from the canonical link provided in the *cache* WNM.
2. The test client should 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.

8.3. 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.

8.3.1. Valid TLS/SSL certificate

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

8.3.2. 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

8.4. Global Discovery Catalogue Service testing

8.4.1. Test setup

The GDC test setup consists of the following:

- A test data bundle to consist of:
  - 100 valid WCMP2 records
  - 1 broken JSON WCMP2 record
  - 1 invalid WCMP2 record
  - 102 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
- all WNM documents updated to point to the correct HTTP server for proper HTTP dereferencing

8.4.2. Functional tests

8.4.2.1. 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:

- `origin/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:
   - `origin/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).

8.4.2.2. 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:
   - `origin/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`)

8.4.2.3. 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:
   - `origin/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`)

8.4.2.4. 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:
   - `origin/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`)

Purpose

The Global Discovery Catalogue should be able to process failing 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:
   - `origin/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 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`)

8.4.2.5. Metadata ingest centre-id mismatch

TODO: validate workflow for regional distribution of partner metadata

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:
    - `origin/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 (3rd token split on `:`).
6. compare the centre-id from the topic and the centre-id of the WCMP2 record identifier.
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`)

8.4.2.6. 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:
    - `origin/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`)

8.4.2.7. 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:
    - `origin/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`)

8.4.2.8. WCMP2 metadata archive zipfile publication

Purpose

Validate that a GDC API publishes a metadata archive zipfile.

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.
8. Initialize the MQTT client with the necessary parameters such as the MQTT protocol version 5, TLS security, and username/password for authentication.
9. Connect the MQTT client to the Global Discovery Catalogue.
10. Once the connection is successful, attempt to subscribe to the following topics:
    - `origin/a/wis2/centre-id/metadata/#` (where `centre-id` is the centre identifier of the Global Discovery Catalogue).
11. Check if the subscription is successful. If the subscription is successful, proceed. If the subscription is not successful, the test fails.
12. If the MQTT client is able to successfully subscribe to the topic on the Global Discovery Catalogue, the test passes. If not, the test fails.
13. On receipt of a notification message, issue a HTTP GET requerst on the canonical link (a link object's `href` property where `rel=canonical`)
14. Unzip the content of the HTTP response.

On successful completion:

- the resulting HTTP response should be a zip encoded data, which, when unzipped, contains a

directory of JSON files of WCMP2 metadata.

8.4.2.9. WCMP2 cold start initialization from metadata archive zipfile

Purpose

Validate that a GDC initializes from a metadata archive zipfile.

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 a Global Replay service endpoint (`https://HOST/collections/wis2-notification-messages?topic=origin/a/wis2/*/metadata&datetime=START_TIME/..`).
    a. `START_TIME` is a timestamp that is from 24 hours ago, in RFC3339 format.
10. Issue a HTTP GET request on the path.
11. Parse the HTTP response.
12. Foreach item in the `features` array:
    a. Check that the item includes a `links` array.
    b. In the `links` array, match the link where `rel=canonical`.
    c. In the matching link, issue a HTTP GET request on the associated `href` value.
    d. Parse the HTTP response.
    e. Validate and ingest into the new GDC
13. Construct a path to the new GDC endpoint (`https://HOST/collections/wis2-discovery-metadata/items?limit=99999`).
14. Issue a HTTP GET request on the path.
15. Parse the HTTP response.
16. Count the numbe rof items in the `features` array.

On successful completion:

- the number of the features in the GDC should match the number of records in the metadata archive zipfile and the number of records from the Global Replay query.

8.4.2.10. OpenMetrics publication

Purpose

Validate that a GDC API publishes an OpenMetrics endpoint.

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=related`, `type=text/plain` and `title=OpenMetrics`.
6. In the matching link, issue a HTTP GET request on the associated `href` value.
7. Parse the HTTP response.

On successful completion:

- the resulting HTTP response should be a text file in OpenMetrics format.

8.4.2.11. 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 greater than 0.
9. Ensure that a `numberReturned` property exists with an integer value greater than 0.
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 greater than 0.
14. Ensure that a `numberReturned` property exists with an integer value greater than 0.
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 greater than 0.
20. Ensure that a `numberReturned` property exists with an integer value greater than 0.
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 greater than 0.
26. Ensure that a `numberReturned` property exists with an integer value greater than 0.
27. Ensure that a `features` array exists.

TODO: measure accuracy

8.4.3. Performance tests

8.4.3.1. 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:
   - `origin/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

Chapter 9. Results

9.1. Global Cache Service results

9.1.1. China

9.1.1.1. Functional test results

9.1.1.2. Performance test results

9.1.1.3. Testing observations

9.1.2. Germany

9.1.2.1. Functional test results

9.1.2.2. Performance test results

9.1.2.3. Testing observations

9.1.3. Japan

9.1.3.1. Functional test results

9.1.3.2. Performance test results

9.1.3.3. Testing observations

9.1.4. Republic of Korea

9.1.4.1. Functional test results

9.1.4.2. Performance test results

9.1.4.3. Testing observations

9.1.5. United Kingdom of Great Britain and Northern Ireland / United States of America

9.1.5.1. Functional test results

9.1.5.2. Performance test results

9.1.5.3. Testing observations

9.2. Global Broker Service results

9.2.1. Brazil

9.2.1.1. Functional test results

9.2.1.2. Performance test results

9.2.1.3. Testing observations

9.2.2. China

9.2.2.1. Functional test results

9.2.2.2. Performance test results

9.2.2.3. Testing observations

9.2.3. France

9.2.3.1. Functional test results

9.2.3.2. Performance test results

9.2.3.3. Testing observations

9.2.4. United States of America

9.2.4.1. Functional test results

9.2.4.2. Performance test results

9.2.4.3. Testing observations

9.3. Global Discovery Catalogue results

9.3.1. Canada

9.3.1.1. Functional test results

9.3.1.2. Performance test results

9.3.1.3. Testing observations

9.3.2. China

9.3.2.1. Functional test results

9.3.2.2. Performance test results

9.3.2.3. Testing observations

9.4. Global Monitor results

9.4.1. China

9.4.1.1. Functional test results

9.4.1.2. Performance test results

9.4.1.3. Testing observations

9.4.2. Morocco

9.4.2.1. Functional test results

9.4.2.2. Performance test results

9.4.2.3. Testing observations

Chapter 10. Discussion

TODO

Chapter 11. Conclusions

TODO

Chapter 12. 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