wis2box

Release 0.5.0

World Meteorological Organization (WMO)

2022-11-15

TABLE OF CONTENTS:

1 Overview **3**
- 1.1 Features . . . 3

2 WIS 2.0 **5**

3 How wis2box works **7**
- 3.1 High level system context . . . 7
- 3.2 Docker Compose . . . 7
- 3.3 Container workflow . . . 7
- 3.4 Technology . . . 10

4 Installation **11**
- 4.1 Requirements and dependencies . . . 11
- 4.2 Installing wis2box . . . 12
- 4.3 Summary . . . 12

5 Quickstart **13**
- 5.1 Requirements and dependencies . . . 13

6 Configuration **15**
- 6.1 Environment variables . . . 15
- 6.2 Sections . . . 15
- 6.3 Docker Compose . . . 19
- 6.4 Summary . . . 19

7 Administration **21**
- 7.1 Public environment variables . . . 21
- 7.2 Default service ports . . . 22
- 7.3 MQTT Quality of Service (QoS) . . . 22

8 Running **23**
- 8.1 Design time . . . 23
- 8.2 Runtime . . . 23
- 8.3 Running topics . . . 23

9 Storage **37**
- 9.1 Overview . . . 37
- 9.2 Uploading data to MinIO . . . 39

10 Monitoring **43**
- 10.1 Prometheus exporters for wis2box . . . 43

10.2 Loki logging . . . 43
10.3 Monitoring topics . . . 44

11 Services **47**
11.1 Discovery Catalogue . . . 47
11.2 Data API . . . 48
11.3 SpatioTemporal Asset Catalog (STAC) . . . 49
11.4 Web Accessible Folder (WAF) . . . 49
11.5 Broker . . . 49
11.6 Adding services . . . 49

12 Authentication and Access control **51**
12.1 Adding Access Control . . . 51
12.2 Authenticating . . . 51
12.3 Removing Access Control . . . 52
12.4 Extending Access Control . . . 52

13 Data access **53**
13.1 Overview . . . 53
13.2 API . . . 53
13.3 PubSub . . . 66
13.4 Running the examples . . . 70
13.5 Summary . . . 71

14 Extending wis2box **73**
14.1 Building your own data plugin . . . 73
14.2 Packaging . . . 74
14.3 Integration . . . 74
14.4 Example plugins . . . 75

15 Development **77**
15.1 Testing . . . 77
15.2 Versioning . . . 77
15.3 Code Conventions . . . 78

16 Contributing **79**

17 Support **81**

18 License **83**
18.1 Software . . . 83
18.2 Documentation . . . 87

19 Indices and tables **89**

Author World Meteorological Organization (WMO)

Contact https://github.com/wmo-im/wis2box

Release 0.5.0

Date 2022-11-15

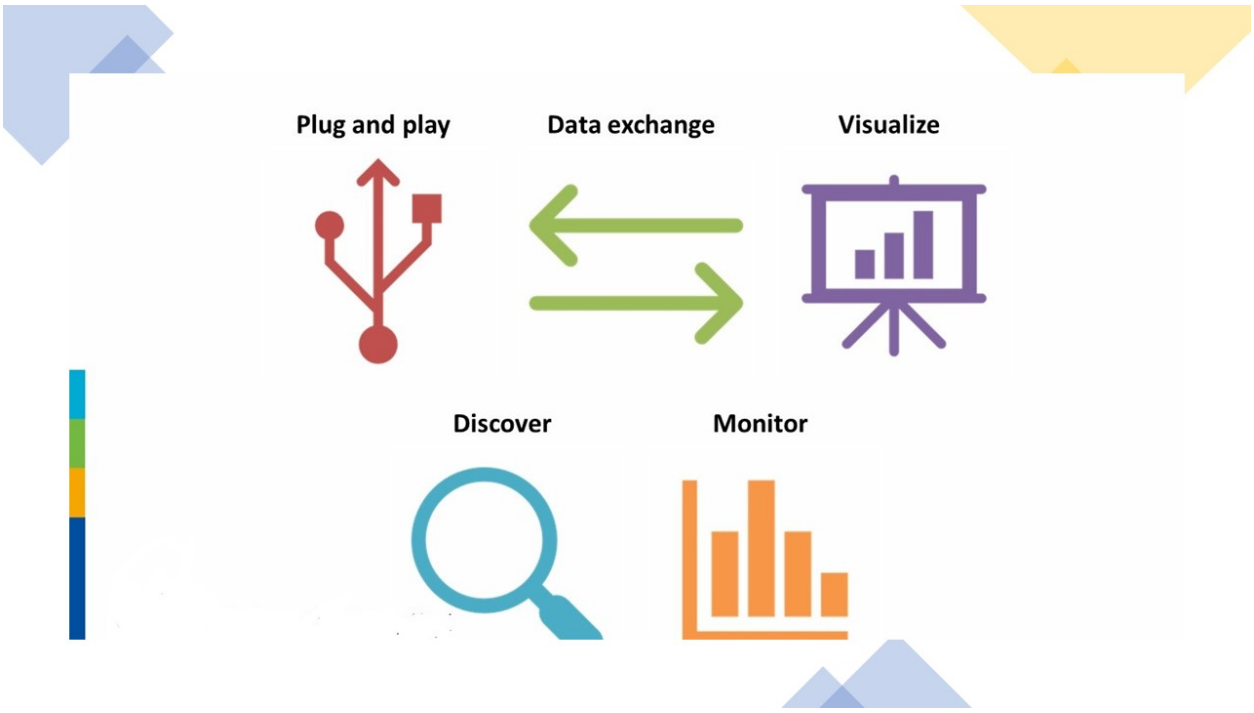
CHAPTER

ONE

OVERVIEW

wis2box is a Python reference implementation of a WMO WIS 2.0 node. The project provides a plug and play toolset to ingest, process, and publish weather/climate/water data using standards-based approaches in alignment with the WIS 2.0 principles. In addition, wis2box also provides access to all data in the WIS 2.0 network, from other wis2box instances and global centres.

wis2box is designed to have a low barrier to entry for data providers, providing enabling infrastructure and services for data discovery, access, and visualization.



1.1 Features

- WIS 2.0 compliant: easily register your wis2box to WIS 2.0 infrastructure, conformant to WMO data and metadata standards
- event driven or interactive data ingest/process/publishing pipeline
- visualization of stations/data on interactive maps
- discovery metadata management and publishing

- download/access of data from the WIS 2.0 network to your local environment
- standards-based data services and access mechanisms:
  - Data formats
    - BUFR
    - GeoJSON
  - Message formats
    - WIS2 Notification Message Format
  - Access and notification protocols
    - HTTP
    - MQTT
  - APIs
    - OGC API
- robust and extensible plugin framework. Write your own data processing engines and integrate seamlessly into wis2box!
- free and open source (FOSS)
- containerized: use of Docker, enabling easy deployment to cloud or on-premises infrastructure

CHAPTER

TWO

WIS 2.0

The WMO Information System is a coordinated global infrastructure responsible for telecommunications and data management functions and is owned and operated by WMO Members.

WIS provides an integrated approach suitable for all WMO Programmes to meet the requirements for routine collection and automated dissemination of observed data and products, as well as data discovery, access, and retrieval services for weather, climate, water, and related data produced by centres and Member countries in the framework of any WMO Programme. It is capable of exchanging large data volumes, such as new ground and satellite-based systems, finer resolutions in numerical weather prediction, and hydrological models and their applications. These data and products must be available to National Hydrological and Meteorological Services (NHMS), but also national disaster authorities for more timely alerts where and when needed.

WIS is a vital data communications backbone for integrating the diverse real-time and non-real-time high priority data sets, regardless of location.

Further documentation on WIS 2.0 can be found at the following links:

- WIS Overview

CHAPTER

THREE

HOW WIS2BOX WORKS

wis2box is implemented in the spirit of the Twelve-Factor App methodology.

wis2box is a Docker and Python-based platform with the capabilities for centres to publish their data holdings to the WMO Information System with a plug and play capability supporting data publishing, discovery and access.

3.1 High level system context

The following diagram provides a high level overview of the main functions of wis2box:

Core wis2box functionality includes the ability to:

- integrate your existing data processing pipeline
- cache station metadata from the OSCAR/Surface station metadata management tool
- process and transform your weather/climate/water data into official WMO data formats
- create and publish discovery metadata of your datasets
- provide your data via OGC and PubSub standards mechanisms to your data, enabling easy access for web applications, desktop GIS tools, mobile applications
- connect your wis2box to the WIS 2.0 network
- make your data and services available to market search engines
- subscribe to and download weather/climate/water data from the WIS 2.0 network

3.2 Docker Compose

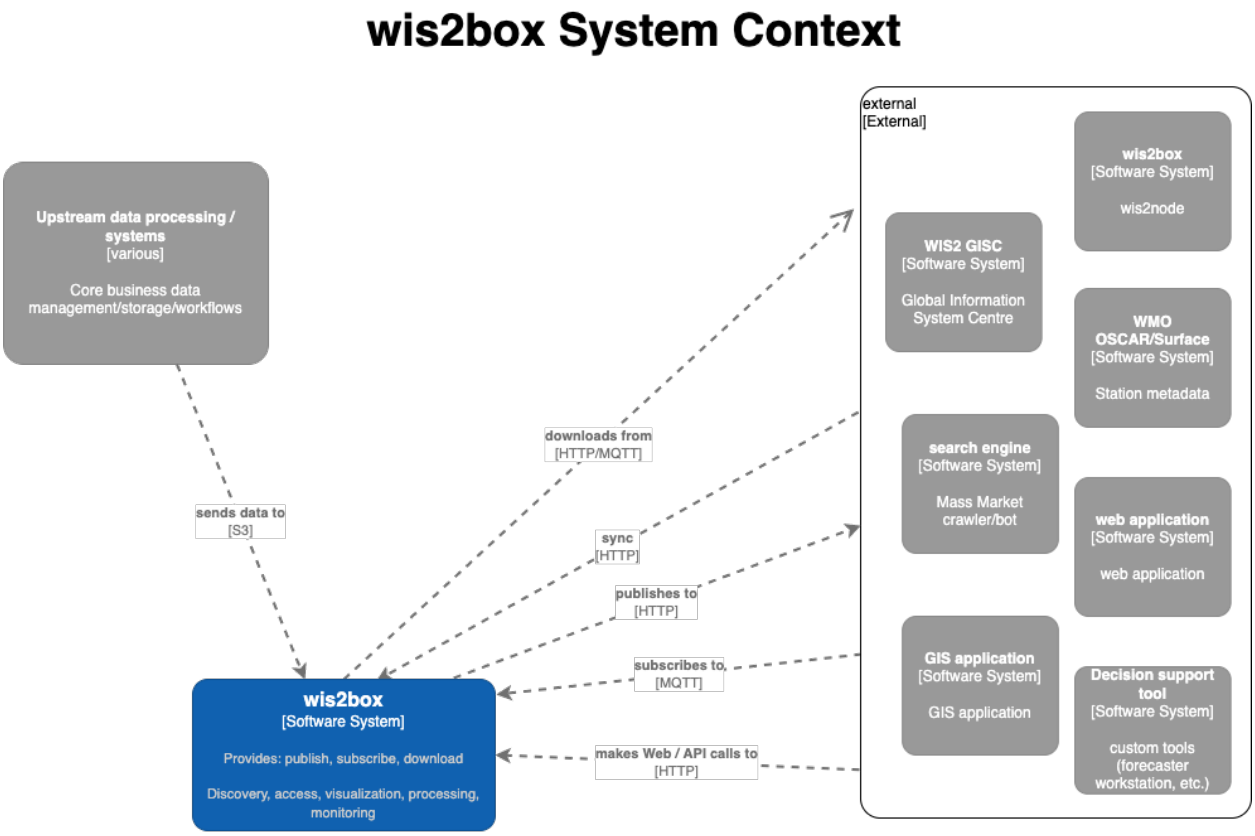
wis2box is built as Docker Compose application, allowing for easy install and container management.

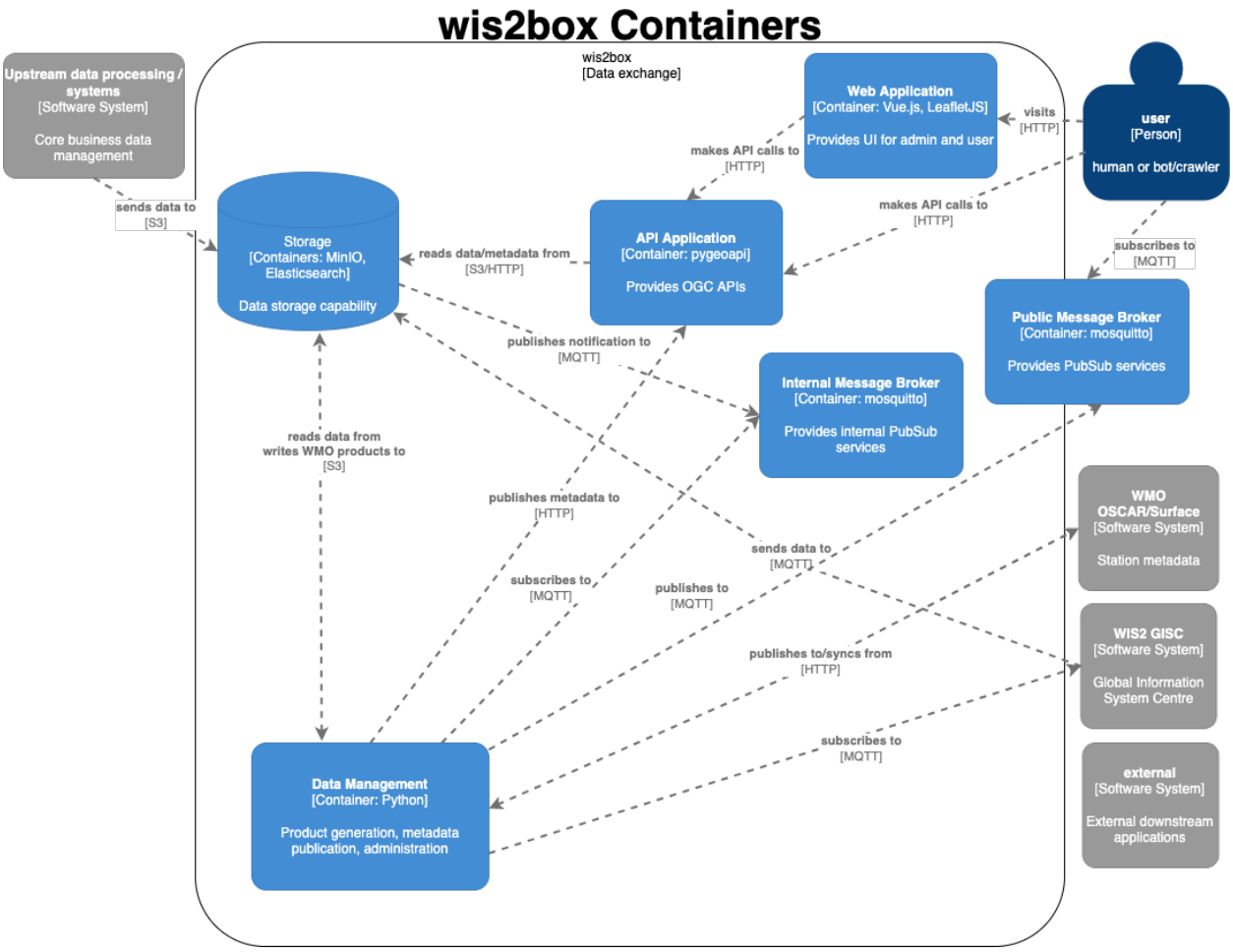
3.3 Container workflow

Let's dive a little deeper. The following diagram provides a view of all wis2box containers:

Container functionality can be described as follows:

- **Storage**: core data and metadata persistence, the initial data entry point of wis2box. Data pipelines and workflow are triggered from here
- **Internal Message Broker**: internal message bus


wis2box System Context
Upstream data processing / systems
[various]
Core business data management/storage/workflows
sends data to
[S3]
wis2box
[Software System]
Provides: publish, subscribe, download
Discovery, access, visualization, processing, monitoring
downloads from
[HTTP/MQTT]
sync
[HTTP]
publishes to
[HTTP]
subscribes to
[MQTT]
makes Web / API calls to
[HTTP]
external
[External]
WIS2 GISC
[Software System]
Global Information System Centre
search engine
[Software System]
Mass Market crawler/bot
GIS application
[Software System]
GIS application
wis2box
[Software System]
wis2node
WMO OSCAR/Surface
[Software System]
Station metadata
web application
[Software System]
web application
Decision support tool
[Software System]
custom tools (forecaster workstation, etc.)


wis2box Containers
wis2box
[Data exchange]
Upstream data processing / systems
[Software System]
Core business data management
sends data to
[S3]
Storage
[Containers: MinIO, Elasticsearch]
Data storage capability
reads data/metadata from
[S3/HTTP]
API Application
[Container: pygeoapi]
Provides OGC APIs
Web Application
[Container: Vue.js, LeafletJS]
Provides UI for admin and user
makes API calls to
[HTTP]
visits
[HTTP]
user
[Person]
human or bot/crawler
makes API calls to
[HTTP]
subscribes to
[MQTT]
Public Message Broker
[Container: mosquitto]
Provides PubSub services
publishes notification to
[MQTT]
Internal Message Broker
[Container: mosquitto]
Provides internal PubSub services
reads data from
writes WMO products to
[S3]
publishes metadata to
[HTTP]
sends data to
[MQTT]
subscribes to
[MQTT]
publishes to
[MQTT]
publishes to/syncs from
[HTTP]
subscribes to
[MQTT]
Data Management
[Container: Python]
Product generation, metadata publication, administration
WMO OSCAR/Surface
[Software System]
Station metadata
WIS2 GISC
[Software System]
Global Information System Centre
external
[Software System]
External downstream applications

- **Public Message Broker**: public facing broker. Provides data and metadata notifications
- **Data Management**: the epicentre of wis2box. Provides core wis2box administration and data/workflow/publishing utilities
- **API Application**: OGC APIs providing geospatial web services
- **Web Application**: user interface

3.4 Technology

wis2box is built on free and open source (FOSS) technology.

Container	Function	Technology	Standards
Storage	data and metadata storage	MinIO Elasticsearch	S3
Internal Message Broker	PubSub	mosquitto	MQTT
Public Message Broker	PubSub	mosquitto	MQTT
Data Management	data processing and publishing	ecCodes csv2bufr bufr2geojson OWSLib pygeometa pyoscar	WCMP (WMO Core Metadata Profile) WMDR (WIGOS Metadata Record)
API Application	data discovery and access	pygeoapi	OGC API
Web Application	data discovery and visualization	Vue.js Leaflet	OGC API

CHAPTER

FOUR

INSTALLATION

wis2box leverages Docker for easy installation across operating systems and environments.

4.1 Requirements and dependencies

Dependencies are installed as containers in the deployment of wis2box. This is true for the wis2box software itself, which runs as a container orchestrating the necessary data management workflows of a node in the WIS 2.0 network.

wis2box requires the following prior to installation:

Requirement	Version
Python	3.8
Docker Engine	20.10.14
Docker Compose	2.4.1

If these are already installed, you can skip to installing wis2box.

- To install Python, follow Python installation.
- To install Docker, follow Docker Engine installation.
- To install Docker Compose, follow Compose installation.

Successful installation can be confirmed by inspecting the versions on your system.

```
docker version
docker compose version
python3 -V
```

Note: Docker may require post-install configuration. Linux users may need to follow post install steps to grant docker privileges. Users in corporate settings my need to configure Docker's HTTP/HTTPS proxy.

4.2 Installing wis2box

Once Python and Docker are installed, the wis2box software needs to be installed.

4.2.1 ZIP archive

wis2box can be installed from a ZIP archive of a the latest branch or a wis2box release.

```
# curl, wget or download from your web browser
curl https://github.com/wmo-im/wis2box/archive/refs/heads/main.zip -L -O -J
unzip wis2box-main.zip
cd wis2box-main
```

4.2.2 GitHub

wis2box can also be installed using the git CLI.

```
# clone wis2box GitHub repository
git clone https://github.com/wmo-im/wis2box.git
cd wis2box
```

4.3 Summary

Congratulations! Whichever of the abovementioned methods you chose, you have successfully installed wis2box onto your system. From here, you can get started with test data by following the *Quickstart*, or continue on to *Configuration*.

CHAPTER

FIVE

QUICKSTART

5.1 Requirements and dependencies

The quickstart assumes wis2box and its dependencies have been installed. If this is not true, please follow the *Installation* steps first.

Successful installation can be confirmed by inspecting the versions on your system.

```
docker version
docker-compose version
python3 -V
```

The quickstart deploys wis2box with test data. It is the minimal runtime configuration profile - as used in wis2box Github CI/CD.

Note: For information on how to quickly get started with your own data out of the box, proceed to *Running*. For more information on deployment, see *Administration* and *Configuration*.

---

wis2box passes environment variables from dev.env to its container on startup. The test enviroment file is provided in `tests/test.env`. Copy this file to `dev.env` in your working directory.

```
cp tests/test.env dev.env
```

Build and update wis2box

```
python3 wis2box-ctl.py build
python3 wis2box-ctl.py update
```

Start wis2box and login to the wis2box container

```
python3 wis2box-ctl.py start
python3 wis2box-ctl.py login
```

Once logged in, verify the enviroment

```
wis2box environment show
```

Publish test discovery metadata

```
wis2box metadata discovery publish $WIS2BOX_DATADIR/metadata/discovery/mwi-surface-
↪weather-observations.yml
wis2box metadata discovery publish $WIS2BOX_DATADIR/metadata/discovery/ita-surface-
↪weather-observations.yml
wis2box metadata discovery publish $WIS2BOX_DATADIR/metadata/discovery/dza-surface-
↪weather-observations.yml
```

Setup observation collections from discovery metadata

```
wis2box data add-collection $WIS2BOX_DATADIR/metadata/discovery/mwi-surface-weather-
↪observations.yml
wis2box data add-collection $WIS2BOX_DATADIR/metadata/discovery/ita-surface-weather-
↪observations.yml
wis2box data add-collection $WIS2BOX_DATADIR/metadata/discovery/dza-surface-weather-
↪observations.yml
```

Ingest data, using data ingest command to push the wis2box-incoming bucket

```
wis2box data ingest --topic-hierarchy mwi.mwi_met_centre.data.core.weather.surface-based-
↪observations.SYNOP --path $WIS2BOX_DATADIR/observations/malawi
wis2box data ingest --topic-hierarchy ita.roma_met_centre.data.core.weather.surface-
↪based-observations.SYNOP --path $WIS2BOX_DATADIR/observations/italy
wis2box data ingest --topic-hierarchy dza.alger_met_centre.data.core.weather.surface-
↪based-observations.SYNOP --path $WIS2BOX_DATADIR/observations/algeria
```

Cache and publish stations

```
wis2box metadata station sync $WIS2BOX_DATADIR/metadata/station/station_list.csv
```

Logout of wis2box container:

```
exit
```

From here, you can run `python3 wis2box-ctl.py status` to confirm that containers are running.

To explore your wis2box installation and services, visit http://localhost:8999 in your web browser.

CHAPTER

SIX

CONFIGURATION

Once you have installed wis2box, it is time to setup the configuration. wis2box setup is based on a simple configuration that can be adjusted depending the user's needs and deployment environment.

6.1 Environment variables

wis2box configuration is driven primarily by a small set of environment variables. The runtime configuration is defined in the Env format in a plain text file named `dev.env` and `docker/default.env`.

Any values set in `dev.env` override the default environment variables in `docker/default.env`. For further / specialized configuration, see the sections below.

6.1.1 WIS2BOX_HOST_DATADIR

The minimum required setting in `dev.env` is the `WIS2BOX_HOST_DATADIR` environment variable. Setting this value is **required** to map the wis2box data directory from the host system to the containers.

It is recommended to set this value to an absolute path on your system.

6.2 Sections

Note: A reference configuration can always be found in the wis2box GitHub repository. The *Quickstart* uses a variant of `wis2box.env` with mappings to the test data, as an example. For complex installations, it is recommended to start configuring wis2box by copying the example `wis2box.env` file and modifying accordingly.

wis2box environment variables can be categorized via the following core sections:

- **Storage**: MinIO configuration
- **API**: API configuration for provisioning the OGC API capabilities
- **Logging**: logging configuaration for wis2box
- **PubSub**: PubSub options
- **Other**: other miscellaneous options

Note: Configuration directives and reference are described below via annotated examples. Changes in configuration require a restart of wis2box to take effect. See the *Administration* section for information on managing wis2box.

6.2.1 Storage

wis2box currently supports S3 compatible storage (e.g. MinIO, Amazon S3). Additional storage types are planned for future releases.

The following environment variables can be used to configure *WIS2BOX_STORAGE*.

Note: When using wis2box in production and using the default MinIO-container, please specify a unique `WIS2BOX_STORAGE_PASSWORD`

```
WIS2BOX_STORAGE_TYPE=S3
WIS2BOX_STORAGE_SOURCE=http://minio:9000
WIS2BOX_STORAGE_USERNAME=minio  # username for the storage-layer
WIS2BOX_STORAGE_PASSWORD=minio123  # password for the storage-layer
WIS2BOX_STORAGE_INCOMING=wis2box-incoming  # name of the storage-bucket/folder for␣
↪incoming files
WIS2BOX_STORAGE_PUBLIC=wis2box-public  # name of the storage-bucket/folder for public␣
↪files
WIS2BOX_STORAGE_ARCHIVE=wis2box-archive  # name of the storage-bucket/folder for␣
↪archived data
WIS2BOX_STORAGE_CONFIG=wis2box-config  # name of the storage-bucket/folder for␣
↪configuration files
WIS2BOX_STORAGE_DATA_RETENTION_DAYS=7  # number of days to keep files in incoming and␣
↪public
```

6.2.2 MinIO

wis2box uses MinIO as the default S3 storage capability.

When overriding the default storage environment variables, please redefine the `MINIO*` environment variables to match your configuration.

```
MINIO_ROOT_USER=${WIS2BOX_STORAGE_USERNAME}
MINIO_ROOT_PASSWORD=${WIS2BOX_STORAGE_PASSWORD}
MINIO_NOTIFY_MQTT_USERNAME_WIS2BOX=${WIS2BOX_BROKER_USERNAME}
MINIO_NOTIFY_MQTT_PASSWORD_WIS2BOX=${WIS2BOX_BROKER_PASSWORD}
MINIO_NOTIFY_MQTT_BROKER_WIS2BOX=tcp://${WIS2BOX_BROKER_HOST}:${WIS2BOX_BROKER_PORT}
```

6.2.3 API

API configurations drive control of the OGC API setup.

```
WIS2BOX_API_TYPE=pygeoapi  # server tpye
WIS2BOX_API_URL=http://localhost:8999/pygeoapi  # public landing page endpoint
WIS2BOX_API_BACKEND_TYPE=Elasticsearch  # backend provider type
WIS2BOX_API_BACKEND_URL=http://elasticsearch:9200  # internal backend connection URL
WIS2BOX_DOCKER_API_URL  # container name of API container (for internal communications/
↪workflow)
```

6.2.4 Logging

The logging directives control logging level/severity and output.

```
WIS2BOX_LOGGING_LOGLEVEL=ERROR  # the logging level (see https://docs.python.org/3/
↪library/logging.html#logging-levels)
WIS2BOX_LOGGING_LOGFILE=stdout  # the full file path to the logfile or ``stdout`` to␣
↪display on console
```

6.2.5 PubSub

PubSub configuration provides connectivity information for the PubSub broker.

```
WIS2BOX_BROKER_HOST=mosquitto  # the hostname of the internal broker
WIS2BOX_BROKER_PORT=1883  # the port of the internal broker
WIS2BOX_BROKER_USERNAME=wis2box  # the username of the internal broker
WIS2BOX_BROKER_PASSWORD=wis2box  # the password of the internal broker
WIS2BOX_BROKER_PUBLIC=mqtt://foo:bar@localhost:1883  # RFC 1738 URL of public broker␣
↪endpoint
```

6.2.6 Web application

Web application configuration provides the ability to customize web components.

```
WIS2BOX_BASEMAP_URL="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"  # URL of map␣
↪tile server to use
WIS2BOX_BASEMAP_ATTRIBUTION="<a href="https://osm.org/copyright">OpenStreetMap</a>␣
↪contributors"  # attribution of map tile server
```

6.2.7 Other

Additional directives provide various configurationscontrol of configuration options for the deployment of wis2box.

```
WIS2BOX_OSCAR_API_TOKEN=some_token  # OSCAR/Surface API token for OSCAR API interaction
WIS2BOX_URL=http://localhost:8999/  # public wis2box url
WIS2BOX_AUTH_STORE=http://wis2box-auth # wis2box auth service location
```

Note: To access internal containers, URL configurations should point to the named containers as specified in `docker-compose.yml`.

A full configuration example can be found below:

```
# please define a data-directory on your host machine
# this will map to /data/wis2box on the wis2box-container
WIS2BOX_HOST_DATADIR==/path/to/local/data/directory

# Optional
# Environment variable overrides
```

(continues on next page)

(continued from previous page)

```
WIS2BOX_LOGGING_LOGLEVEL=WARNING
WIS2BOX_DATA_RETENTION_DAYS=30

WIS2BOX_DATADIR_DATA_MAPPINGS=/data/wis2box/data-mappings.yml
```

```
# data paths and retention
WIS2BOX_DATADIR=/data/wis2box

# API
WIS2BOX_API_TYPE=pygeoapi
WIS2BOX_API_URL=http://localhost:8999/oapi
WIS2BOX_API_BACKEND_TYPE=Elasticsearch
WIS2BOX_API_BACKEND_URL=http://elasticsearch:9200
WIS2BOX_DOCKER_API_URL=http://wis2box-api:80/oapi

# logging
WIS2BOX_LOGGING_LOGLEVEL=ERROR
WIS2BOX_LOGGING_LOGFILE=stdout

# PubSub
WIS2BOX_BROKER_USERNAME=wis2box
WIS2BOX_BROKER_PASSWORD=wis2box
WIS2BOX_BROKER_HOST=mosquitto
WIS2BOX_BROKER_PORT=1883

WIS2BOX_BROKER_PUBLIC=mqtt://wis2box:wis2box@mosquitto:1883

# Web application
WIS2BOX_BASEMAP_URL=https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png
WIS2BOX_BASEMAP_ATTRIBUTION=<a href="https://osm.org/copyright">OpenStreetMap</a>␣
↪contributors

# other
WIS2BOX_OSCAR_API_TOKEN=some_token
WIS2BOX_URL=http://localhost:8999

# mappings of topic hierarchy to wis2box data plugins
# optionally override default mappings from wis2box data plugins
# WIS2BOX_DATADIR_DATA_MAPPINGS=${PWD}/data-mappings.yml

# access control
WIS2BOX_AUTH_URL=http://wis2box-auth

# storage
WIS2BOX_STORAGE_TYPE=S3
WIS2BOX_STORAGE_SOURCE=http://minio:9000
WIS2BOX_STORAGE_USERNAME=minio
WIS2BOX_STORAGE_PASSWORD=minio123
WIS2BOX_STORAGE_INCOMING=wis2box-incoming
WIS2BOX_STORAGE_PUBLIC=wis2box-public
WIS2BOX_STORAGE_ARCHIVE=wis2box-archive
WIS2BOX_STORAGE_CONFIG=wis2box-config
```

(continues on next page)

(continued from previous page)

```
WIS2BOX_STORAGE_DATA_RETENTION_DAYS=7

# you should be okay from here

# MinIO
MINIO_ROOT_USER=${WIS2BOX_STORAGE_USERNAME}
MINIO_ROOT_PASSWORD=${WIS2BOX_STORAGE_PASSWORD}
MINIO_PROMETHEUS_AUTH_TYPE=public
MINIO_NOTIFY_MQTT_ENABLE_WIS2BOX=on
MINIO_NOTIFY_MQTT_USERNAME_WIS2BOX=${WIS2BOX_BROKER_USERNAME}
MINIO_NOTIFY_MQTT_PASSWORD_WIS2BOX=${WIS2BOX_BROKER_PASSWORD}
MINIO_NOTIFY_MQTT_BROKER_WIS2BOX=tcp://${WIS2BOX_BROKER_HOST}:${WIS2BOX_BROKER_PORT}
MINIO_NOTIFY_MQTT_TOPIC_WIS2BOX=wis2box-storage/minio
MINIO_NOTIFY_MQTT_TOPIC_WIS2BOX=wis2box-storage/minio
MINIO_NOTIFY_MQTT_QOS_WIS2BOX=1
```

6.3 Docker Compose

The Docker Compose setup is driven from the resulting `dev.env` file created. For advanced cases and/or power users, updates can also be made to `docker-compose.yml` or `docker-compose.override.yml` (for changes to ports).

6.4 Summary

At this point, you have defined the runtime configuration required to administer your wis2box installation.

CHAPTER

SEVEN

ADMINISTRATION

wis2box is designed to be built as a network of virtual machines within a virtual network. Once this is built, users login into the main wis2box machine to setup their workflow and configurations for data processing and publishing.

The `wis2box-ctl.py` utility provides a number of tools for managing the wis2box containers.

The following steps provide an example of container management workflow.

```
# build all images
python3 wis2box-ctl.py build

# start system
python3 wis2box-ctl.py start

# stop system
python3 wis2box-ctl.py stop

# view status of all deployed containers
python3 wis2box-ctl.py status
```

Note: Run `python3 wis2box-ctl.py --help` for all usage options.

With wis2box now installed and started, it's time to start up the box and login to the wis2box container:

```
python3 wis2box-ctl.py start
python3 wis2box-ctl.py login
```

Now that you are logged into the wis2box container, it's now time to manage station metadata, discovery metadata and data processing pipelines.

7.1 Public environment variables

The following environment variables are used for public services:

- `WIS2BOX_API_URL`: API application
- `WIS2BOX_BROKER_PUBLIC`: MQTT broker
- `WIS2BOX_URL`: Web application, including access to data download/object storage

7.2 Default service ports

A default wis2box installation utilizes the following ports for public services:

7.2.1 Public services

- `8999`: Web application, API application, storage
- `1883`: Message broker via MQTT
- `8884`: Message broker via MQTT/WebSockets

7.2.2 Internal services

- `1883`: Message broker
- `9200`: Elasticsearch
- `9000`: MinIO
- `9001`: MinIO admin UI

7.2.3 Changing default ports

The `docker/docker-compose.override.yml` file provides definitions on utilized ports. To change default ports, edit `docker/default.env` before stopping and starting wis2box for changes to take effect.

7.3 MQTT Quality of Service (QoS)

The quality of service level of all wis2box powered brokers is always 1 by default.

CHAPTER

EIGHT

RUNNING

wis2box workflows can be categorized as design time (interactive) or runtime (automated).

8.1 Design time

- environment creation
- topic hierarchy registration
- station metadata caching
- station metadata API publishing
- discovery metadata API publishing

8.2 Runtime

- automated data processing and publishing

8.3 Running topics

8.3.1 Concepts

Let's clarify a few concepts as part working with wis2box:

- **topic hierarchy**: structure defined by WMO to categorize and classify data, allowing for easy and efficient search and identification
- **discovery metadata**: description of a dataset to be included in the WIS 2.0 Global Discovery Catalogue
- **catalogue**: a collection of discovery metadata records
- **station metadata**: description of the properties of an observing station, which provides observations and measurements
- **data mappings**: the wis2box mechanism to define and associate a topic hierarchy to a processing pipeline

8.3.2 Topic hierarchy

Note: The WIS 2.0 topic hierarchy is currently in development. wis2box implementation of the topic hierarchies will change, based on ratifications/updates of the topic hierarchies in WMO technical regulations and publications.

wis2box implements the WIS 2.0 topic hierarchies, which are designed to efficiently categorize and classify data.

8.3.3 Environment

wis2box initializes the environment when starting, before data processing or publishing. To view the environment, run the following command:

```
wis2box environment show
```

For the purposes of documentation, the value `WIS2BOX_DATADIR` represents the base directory for all data managed in wis2box.

The default enviroment variables are below.

```
# data paths and retention
WIS2BOX_DATADIR=/data/wis2box

# API
WIS2BOX_API_TYPE=pygeoapi
WIS2BOX_API_URL=http://localhost:8999/oapi
WIS2BOX_API_BACKEND_TYPE=Elasticsearch
WIS2BOX_API_BACKEND_URL=http://elasticsearch:9200
WIS2BOX_DOCKER_API_URL=http://wis2box-api:80/oapi

# logging
WIS2BOX_LOGGING_LOGLEVEL=ERROR
WIS2BOX_LOGGING_LOGFILE=stdout

# PubSub
WIS2BOX_BROKER_USERNAME=wis2box
WIS2BOX_BROKER_PASSWORD=wis2box
WIS2BOX_BROKER_HOST=mosquitto
WIS2BOX_BROKER_PORT=1883

WIS2BOX_BROKER_PUBLIC=mqtt://wis2box:wis2box@mosquitto:1883

# Web application
WIS2BOX_BASEMAP_URL=https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png
WIS2BOX_BASEMAP_ATTRIBUTION=<a href="https://osm.org/copyright">OpenStreetMap</a>␣
↪contributors

# other
WIS2BOX_OSCAR_API_TOKEN=some_token
WIS2BOX_URL=http://localhost:8999

# mappings of topic hierarchy to wis2box data plugins
# optionally override default mappings from wis2box data plugins
```

(continues on next page)

(continued from previous page)

```
# WIS2BOX_DATADIR_DATA_MAPPINGS=${PWD}/data-mappings.yml

# access control
WIS2BOX_AUTH_URL=http://wis2box-auth

# storage
WIS2BOX_STORAGE_TYPE=S3
WIS2BOX_STORAGE_SOURCE=http://minio:9000
WIS2BOX_STORAGE_USERNAME=minio
WIS2BOX_STORAGE_PASSWORD=minio123
WIS2BOX_STORAGE_INCOMING=wis2box-incoming
WIS2BOX_STORAGE_PUBLIC=wis2box-public
WIS2BOX_STORAGE_ARCHIVE=wis2box-archive
WIS2BOX_STORAGE_CONFIG=wis2box-config
WIS2BOX_STORAGE_DATA_RETENTION_DAYS=7

# you should be okay from here

# MinIO
MINIO_ROOT_USER=${WIS2BOX_STORAGE_USERNAME}
MINIO_ROOT_PASSWORD=${WIS2BOX_STORAGE_PASSWORD}
MINIO_PROMETHEUS_AUTH_TYPE=public
MINIO_NOTIFY_MQTT_ENABLE_WIS2BOX=on
MINIO_NOTIFY_MQTT_USERNAME_WIS2BOX=${WIS2BOX_BROKER_USERNAME}
MINIO_NOTIFY_MQTT_PASSWORD_WIS2BOX=${WIS2BOX_BROKER_PASSWORD}
MINIO_NOTIFY_MQTT_BROKER_WIS2BOX=tcp://${WIS2BOX_BROKER_HOST}:${WIS2BOX_BROKER_PORT}
MINIO_NOTIFY_MQTT_TOPIC_WIS2BOX=wis2box-storage/minio
MINIO_NOTIFY_MQTT_TOPIC_WIS2BOX=wis2box-storage/minio
MINIO_NOTIFY_MQTT_QOS_WIS2BOX=1
```

8.3.4 Data mappings

Once a topic hierarchy is defined, it needs to be included in the wis2box data mappings configuration. wis2box provides a default data mapping (in YAML format):

```
data:
    mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP:
        plugins:
            csv:
                - plugin: wis2box.data.csv2bufr.ObservationDataCSV2BUFR
                  template: synop-bufr.json
                  notify: true
                  file-pattern: '^WIGOS_(\d-\d+-\d+-\w+)_.*\.csv$'
            bufr4:
                - plugin: wis2box.data.bufr2geojson.ObservationDataBUFR2GeoJSON
                  file-pattern: '^WIGOS_(\d-\d+-\d+-\w+)_.*\.bufr4$'
    ita.roma_met_centre.data.core.weather.surface-based-observations.SYNOP:
        plugins:
            bin:
                - plugin: wis2box.data.bufr4.ObservationDataBUFR
                  notify: true
```

(continues on next page)

(continued from previous page)

```
                file-pattern: '^.*\.bin$'
            bufr4:
                - plugin: wis2box.data.bufr2geojson.ObservationDataBUFR2GeoJSON
                  file-pattern: '^WIGOS_(\d-\d+-\d+-\w+)_.*\.bufr4$'
    dza.alger_met_centre.data.core.weather.surface-based-observations.SYNOP:
        plugins:
            bufr4:
                - plugin: wis2box.data.bufr4.ObservationDataBUFR
                  notify: true
                  buckets:
                    - ${WIS2BOX_STORAGE_INCOMING}
                  file-pattern: '^.*\.bufr4$'
                - plugin: wis2box.data.bufr2geojson.ObservationDataBUFR2GeoJSON
                  buckets:
                    - ${WIS2BOX_STORAGE_PUBLIC}
                  file-pattern: '^WIGOS_(\d-\d+-\d+-\w+)_.*\.bufr4$'
```

The data mappings are indicated by the `data` keyword, with each topic having a separate entry specifying:

- `plugins`: all plugin objects associated with the topic, by file type/extension

Each plugin is based on the file extension to be detected and processed, with the following configuration:

- `plugin`: the codepath of the plugin
- `notify`: whether the plugin should publish a data notification
- `template`: additional argument allowing a mapping template name to be passed to the plugin
- `file-pattern`: additional argument allowing a file pattern to be passed to the plugin
- `buckets`: the name(s) of the storage bucket(s) that data should be saved to (See *Configuration* for more informaiton on buckets)

The default data mapping can be overriden by user-defined data mappings with the following steps:

- create a YAML file similar to the above to include your topic hierarchy
- set the `WIS2BOX_DATADIR_DATA_MAPPINGS` environment variable to point to the new file of definitions
- restart wis2box

See *Extending wis2box* for more information on adding your own data processing pipeline.

8.3.5 Station metadata

wis2box is designed to support data ingest and processing of any kind. For observations, processing workflow typically requires station metadata to be present at runtime.

wis2box provides the ability to cache station metadata from the WMO OSCAR/Surface system.

To cache your stations of interest, create a CSV file formatting per below, specifying one line (with station name and WIGOS station identifier [WSI]) per station:

```
station_name,wigos_station_identifier,traditional_station_identifier
Balaka,0-454-2-AWSBALAKA,
Malomo_EPA,0-454-2-AWSMALOMO,
IN-GUEZZAM,0-20000-0-60690,60690
BENI OUNIF,0-12-0-08BECCN60577,60577
```

Use this CSV to cache station metadata:

```
wis2box metadata station cache /path/to/station_list.csv
```

Resulting station metadata files (JSON) are stored in `WIS2BOX_DATADIR/data/metadata/station` and can be used by wis2box data processing pipelines. These data are required before starting automated processing.

Note: run the command `wis2box metadata station sync /path/to/station_list.csv` to both cache stations form OSCAR/Surface and publish station as a collection to the wis2box API

See also:

API publishing

Summary

At this point, you have cached the required station metadata for your given dataset(s).

8.3.6 Discovery metadata

Discovery metadata describes a given dataset or collection. Data being published through a wis2box requires discovery metadata (describing it) to be created, maintained and published to the wis2box catalogue API.

wis2box supports managing discovery metadata using the WMO Core Metadata Profile (WCMP) 2.0 standard.

Note: WCMP 2.0 is currently in development as part of WMO activities.

Creating a discovery metadata record in wis2box is as easy as completing a YAML configuration file. wis2box leverages the pygeometa project's metadata control file (MCF) format. Below is an example MCF file.

```
wis2box:
    retention: P30D
    topic_hierarchy: mwi.mwi_met_centre.data.core.weather.surface-based-observations.
↪SYNOP
    data_category: observationsSurfaceLand
    country: mwi
    center_id: mwi_met_centre

mcf:
    version: 1.0

metadata:
    identifier: mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP
    language: en
    language_alternate: fr
    charset: utf8
    hierarchylevel: dataset
    datestamp: 2021-11-29

spatial:
    datatype: vector
```

(continues on next page)

(continued from previous page)

```
    geomtype: point

identification:
    language: en
    charset: utf8
    title:
        en: Surface weather observations from Malawi
    abstract:
        en: Surface weather observations from Malawi
    dates:
        creation: 2021-11-29
        publication: 2021-11-29
    keywords:
        default:
            keywords:
                en:
                    - surface weather
                    - temperature
                    - observations
        wmo:
            keywords:
                en:
                    - weatherObservations
            keywords_type: theme
            vocabulary:
                name:
                    en: WMO Category Code
                url: https://github.com/wmo-im/wcmp-codelists/blob/main/codelists/WMO_
↪CategoryCode.csv
        wis2:
            keywords:
                en:
                    - mw.malawi.weatherObservations.dataset_name
            keywords_type: theme
            vocabulary:
                name:
                    en: WMO Core Metadata profile topic hierarchy
                url: https://github.com/wmo-im/wcmp2-codelists/blob/main/codelists/topic_
↪hierarchy.csv

    topiccategory:
        - climatologyMeteorologyAtmosphere
    extents:
        spatial:
            - bbox: [32.6881653175,-16.8012997372,35.7719047381,-9.23059905359]
              crs: 4326
        temporal:
            - begin: 2021-11-29
              end: null
              resolution: P1H
    fees: None
    accessconstraints: otherRestrictions
```

(continues on next page)

(continued from previous page)

```
    rights:
        en: WMO Unified Policy for the International Exchange of Earth System Data
    url: https://example.org/malawi-surface-weather-observations
    status: onGoing
    maintenancefrequency: continual

contact:
    pointOfContact: &contact_poc
        organization: Department of Climate Change and Meteorologial Services (DCCMS)
        url: https://www.metmalawi.gov.mw
        individualname: Firstname Lastname
        positionname: Position Name
        phone: +265-1-822-014
        fax: +265-1-822-215
        address: P.O. Box 1808
        city: Blantyre
        administrativearea: Blantyre District
        postalcode: M3H 5T4
        country: Malawi
        email: you@example.org
        hoursofservice: 0700h - 1500h UTC
        contactinstructions: email

    distributor: *contact_poc

dataquality:
    scope:
        level: dataset
    lineage:
        statement: this data was generated by the csv2bufr tool
```

Note: There are no conventions to the MCF filename. The filename does not get used/exposed or published. It is up to the user to determine the best filename, keeping in mind your wis2box system may manage and publish numerous datasets (and MCF files) over time.

Summary

At this point, you have created discovery metadata for your given dataset(s).

8.3.7 Data ingest, processing and publishing

At this point, the system is ready for ingest/processing and publishing.

Data ingest, processing and publishing can be run in automated fashion or via the wis2box CLI. Data is ingested, processed, and published as WMO BUFR data, as well as GeoJSON features.

Warning: GeoJSON **data** representations provided in wis2box are in development and are subject to change based on evolving requirements for observation data representations in WIS 2.0 technical regulations.

Interactive ingest, processing and publishing

The *wis2box* CLI provides a data subsystem to process data interactively. CLI data ingest/processing/publishing can be run with explicit or implicit topic hierarchy routing (which needs to be tied to the pipeline via the *Data mappings*).

Explicit topic hierarchy workflow

```
# process a single CSV file
wis2box data ingest --topic-hierarchy foo.bar.baz -p /path/to/file.csv

# process a directory of CSV files
wis2box data ingest --topic-hierarchy foo.bar.baz -p /path/to/dir

# process a directory of CSV files recursively
wis2box data ingest --topic-hierarchy foo.bar.baz -p /path/to/dir -r
```

Implicit topic hierarchy workflow

```
# process incoming data; topic hierarchy is inferred from fuzzy filepath equivalent
# wis2box will detect 'foo/bar/baz' as topic hierarchy 'foo.bar.baz'
wis2box data ingest -p /path/to/foo/bar/baz/data/file.csv
```

Event driven ingest, processing and publishing

Once all metadata and topic hierarchies are setup, event driven workflow will immediately start to listen on files in the `wis2box-incoming` storage bucket as they are placed in the appropriate topic hierarchy directory.

Summary

Congratulations! At this point, you have successfully setup a wis2box data pipeline. Data should be flowing through the system.

8.3.8 Data pipeline plugins

Driven by topic hierarchies, wis2box is a plugin architecture orchestrating all the required components of a WIS 2.0 node. wis2box also provides a data pipeline plugin architecture which allows for users to define a plugin based on a topic hierarchy to publish incoming data (see *Data mappings* for more information).

See also:

Extending wis2box

See also:

Data mappings

Default pipeline plugins

wis2box provides a number of data pipeline plugins which users can be used "out of the box".

`wis2box.data.csv2bufr.ObservationDataCSV2BUFR`

This plugin converts CSV observation data into BUFR using `csv2bufr`. A csv2bufr template can be configured to process the data accordingly. In addition, `file-pattern` can be used to filter on incoming data based on a regular expression. Consult the csv2bufr documentation for more information on configuration and templating.

`wis2box.data.bufr4.ObservationDataBUFR2GeoJSON`

This plugin converts BUFR observation data into GeoJSON using `bufr2geojson`. A `file-pattern` can be used to filter on incoming data based on a regular expression. Consult the bufr2geojson documentation for more information on configuration and templating.

`wis2box.data.geojson.ObservationDataGeoJSON`

This plugin is for the purposes of publishing GeoJSON data to the API.

8.3.9 API publishing

When wis2box starts, the API provisioning environment is initialized. At this stage, the following steps are required:

- station metadata has been configured
- discovery metadata has been created
- data pipelines are configured and running

Let's dive into publishing the data and metadata:

wis2box provides an API supporting the OGC API standards using pygeoapi.

Station metadata API publishing

The first step is to publish our station metadata to the API. The command below will generate local station collection GeoJSON for pygeoapi publication.

```
wis2box metadata station publish-collection
```

Note: run the command `wis2box metadata station sync /path/to/station_list.csv` to both cache stations from OSCAR/Surface and publish station as a collection to the wis2box API

See also:

Station metadata

Discovery metadata API publishing

This step will publish dataset discovery metadata to the API.

```
wis2box metadata discovery publish /path/to/discovery-metadata.yml
```

Dataset collection API publishing

The below command will add the dataset collection to pygeoapi from the discovery metadata MCF created as described in the *Discovery metadata* section.

```
wis2box data add-collection $WIS2BOX_DATADIR/data/config/foo/bar/baz/discovery-metadata.
↪yml
```

To delete the colection from the API backend and configuration:

```
wis2box api delete-collection foo.bar.baz
```

Note: Changes to the API configuration are reflected and updated automatically.

Summary

At this point, you have successfully published the required data and metadata collections to the API.

8.3.10 Data retention

wis2box is configured to set data retention according to your requirements. Data retention is managed via the `WIS2BOX_STORAGE_DATA_RETENTION_DAYS` environment variable as part of configuring wis2box.

Cleaning

Cleaning applies to storage defined by `WIS2BOX_STORAGE_PUBLIC` and involves the deletion of files after set amount of time.

Cleaning is performed by default daily at 0Z by the system, and can also be run interactively with:

```
# delete data older than WIS2BOX_STORAGE_DATA_RETENTION_DAYS by default
wis2box data clean


# delete data older than --days (force override)
wis2box data clean --days=30
```

Archiving

Archiving applies to storage defined by `WIS2BOX_STORAGE_INCOMING` and involves moving files to the storage defined by `WIS2BOX_STORAGE_ARCHIVE`.

Archive is performed on incoming data by default daily at 1Z by the system, and can also be run interactively with:

```
wis2box data archive
```

Only files with a timestamp older than one hour are considered for archiving.

8.3.11 Too Long Didn't Read

For the truly impatient, this section summarizes the steps to required to run wis2box from example configurations.

Environment

wis2box passes environment variables from dev.env to its containers on startup. An example file is provided in `examples/config/wis2box.env`. Copy this file to your working directory, and update it to suit your needs.

```
cp examples/config/wis2box.env dev.env
```

Note: You must map `WIS2BOX_HOST_DATADIR` to the absolute path of directory on your host machine.

For terminals that support environment variables, it may be useful to also define environment variables locally. For terminals that do not support environment variables, you will need to define the asolute path to your data directory.

```
# source
. dev.env
# verify
echo $WIS2BOX_HOST_DATADIR
```

Data mappings

To start processing your own data, you need to define a data mappings file. Example mapping files are provided in examples/config:

- `examples/config/synop-bufr-mappings.yml`, input is .bufr containing SYNOP observation-data
- `examples/config/synop-csv-mappings.yml`, input is .csv containing SYNOP observation-data

To publish .bufr files of SYNOP data:

```
cp examples/config/synop-bufr-mappings.yml $WIS2BOX_HOST_DATADIR/data-mappings.yml
```

Edit `$WIS2BOX_HOST_DATADIR/data-mappings.yml` and change `[ISO3C_country].[center_id].data.core.weather.surface-based-observations.SYNOP`:

- replace `ISO3C_country` with your corresponding ISO 3166 alpha-3 code.
- replace `center_id` with the string identifying the center running the wis2node.

Station metadata

wis2box needs to have a station_list.csv that contains the stations that will be sending data. An example is provided in `examples/config/station_list.csv`. Update the file with your stations.

```
cp examples/config/station_list.csv $WIS2BOX_HOST_DATADIR/station_list.csv
```

Discovery metadata

Discovery metadata is central to wis2box, wis2box-api, and wis2box-ui. An example is provided in `examples/config/surface-weather-observations.yml`.

```
cp examples/config/surface-weather-observations.yml $WIS2BOX_HOST_DATADIR/surface-
↪weather-observations.yml
```

Edit the file `$WIS2BOX_HOST_DATADIR/surface-weather-observations.yml` to provide the correct metadata for your dataset:

- replace `[ISO3C_country].[center_id].data.core.weather.surface-based-observations.SYNOP` with the topic you used in `$WIS2BOX_HOST_DATADIR/data-mappings.yml` previously
- text provided in title and abstract will be displayed in wis2box-ui
- provide a valid bounding-box in bbox

Build wis2box

Run the `build` and `update` commands to set up wis2box. This will start the process of building the wis2box containers from source.

```
python3 wis2box-ctl.py build
python3 wis2box-ctl.py update
```

This might take a while the first time.

Start wis2box

Start wis2box containers and check that all services are running (and healthy).

```
python3 wis2box-ctl.py start
python3 wis2box-ctl.py status
```

If neccessary repeat the command until all services are up and running.

Runtime configuration

The last design-time steps required to run wis2box are once wis2box is running.

Login to the wis2box container

```
python3 wis2box-ctl.py login
```

Note: $WIS2BOX_DATADIR is the location that $WIS2BOX_HOST_DATADIR binds to the container. This allows wis2box command to access the configuration files from inside the wis2box container.

Setup observation data processing and API publication:

```
wis2box data add-collection $WIS2BOX_DATADIR/surface-weather-observations.yml
```

Cache and publish station collection and discovery metadata to the API:

```
wis2box metadata discovery publish $WIS2BOX_DATADIR/surface-weather-observations.yml
wis2box metadata station sync $WIS2BOX_DATADIR/station_list.csv
```

Logout of wis2box container:

```
exit
```

Data ingest

The runtime component of wis2box is data ingestion. This is an event driven workflow driven by s3 notifications from uploading data to wis2box-storage. An example is provided in examples/scripts/copy_to_incoming.py. To access the storage component, visit http://localhost:9001 in your web browser. The default username/password is minio/minio123

Debugging

Something's now working? wis2box includes a local grafana-instance to help you collect and view logs and figure out what's wrong. Visit http://localhost:3000 in your local web browser to view the local grafana instance.

wis2box-ui

wis2box includes a UI to view the data that has been ingested. To explore, visit http://localhost:8999 in your web browser.

Not seeing data? After data has been ingested for a station for the first time, you need to re-publish the stations. This will republish the station with a link relation to any associated observation collection.

```
python3 wis2box-ctl.py execute wis2box metadata station publish-collection
```

CHAPTER

NINE

STORAGE

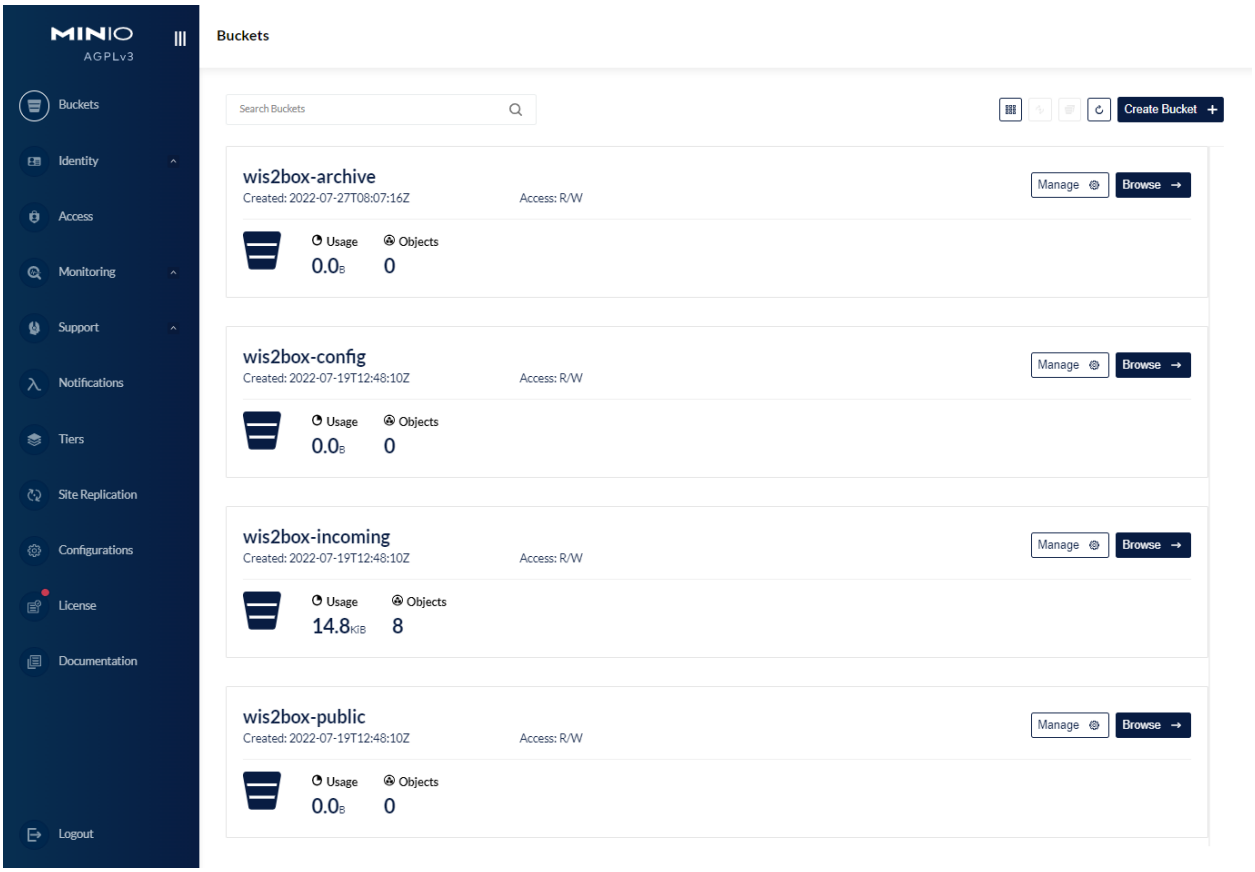
9.1 Overview

The default wis2box storage capability is powered by MinIO, which provides S3 compatible object storage.

The default wis2box MinIO administration user interface can be accessed locally at `http://localhost:9001`.

The username/password for MinIO is configured through environment variables (see *Configuration*).

Once logged in, buckets can be managed via the default "Buckets" menu item (click "Manage"). Click "Browse" provides a browsing capability for a storage administrator.



9.2 Uploading data to MinIO

Files can uploaded to the MinIO bucket in a number of ways. Any new file received on MinIO will trigger an MQTT notification which is received by wis2box.

Below are basic examples on sending data to the MinIO `wis2box-incoming` bucket. For more information and additional examples, consult the official MinIO documentation.

9.2.1 Using the boto3 Python Client

Install the Python boto3 package via pip:

```
pip3 install boto3
```

The below example copies a local file (`myfile.csv`) to the `wis2box-incoming` bucket with topic `foo.bar.baz`:

```
import boto3

endpoint_url = '<your-wis2box-url>'
filename = 'myfile.csv'

session = boto3.Session(
    aws_access_key_id='wis2box',
```

(continues on next page)

(continued from previous page)

```
    aws_secret_access_key='Wh00data!'
)

s3client = session.client('s3', endpoint_url=endpoint_url)

with open(filename, 'rb') as fh:
    s3client.upload_fileobj(fh, 'wis2box-incoming', f'foo/bar/baz/{filename}')
```

To allow uploading files into MinIO remotely, the `wis2box-incoming` bucket is proxied via Nginx.

For example, to upload the local file (`WIGOS_0-454-2-AWSNAMITAMBO_2021-11-18T0955.csv with topic`) with topic `mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP` via the Nbinx proxy:

```
import boto3

endpoint_url = 'http://localhost:9000'
filename = 'myfile.csv'

session = boto3.Session(
    aws_access_key_id='wis2box',
    aws_secret_access_key='Wh00data!'
)

s3client = session.client('s3', endpoint_url=endpoint_url)

filename = 'WIGOS_0-454-2-AWSNAMITAMBO_2021-11-18T0955.csv'

with open(filename, 'rb') as f:
    s3client.upload_fileobj(f, 'wis2box-incoming', f'data/core/observations-surface-land/
↪mw/FWCL/landFixed/{filename}')
```

9.2.2 Using the MinIO Python Client

MinIO provides a Python client which can be used as follows:

Install the Python minio module via pip:

```
pip3 install minio
```

The below example copies a local file (`myfile.csv`) to the `wis2box-incoming` bucket to topic `foo.bar.baz`:

```
from minio import Minio

client = Minio(
    'localhost:9000',
    access_key='minio',
    secret_key='minio123',
    secure=False
)

client.fput_object('wis2box-incoming', 'myfile.csv', '/foo/bar/baz/myfile.csv')
```

9.2.3 Using S3cmd

Given MinIO is S3 compatible, data can be uploaded using generic S3 tooling. The below example uses S3cmd to upload data to wis2box MinIO storage:

Edit the following fields in `~/.s3cfg`:

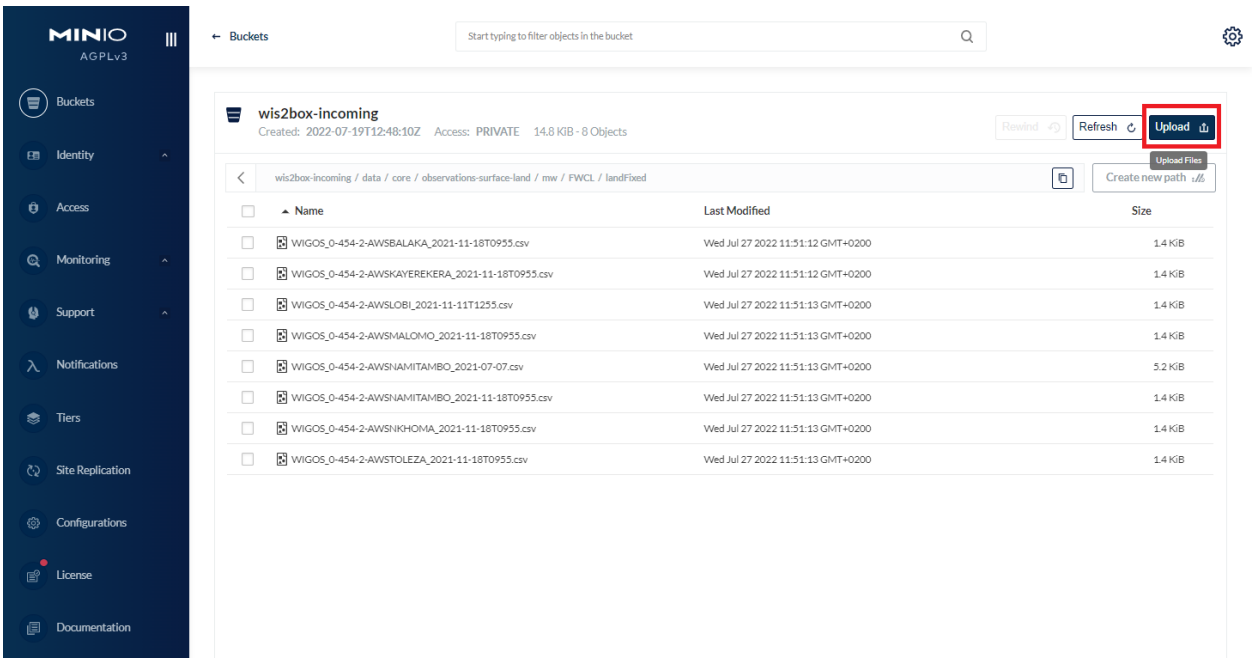
```
cat << EOF > ~/.s3cfg
# Setup endpoint
host_base = localhost:9000
use_https = False

# Setup access keys
access_key = minio
secret_key = minio123
EOF
```

Below is a simple command line example to copy a local file called `myfile.csv` into the `wis2box-incoming` bucket, to topic `foo/bar/baz`:

```
s3cmd myfile.csv s3://wis2box-incoming/foo/bar/baz
```

9.2.4 Using the MinIO UI

Files can also be uploaded interactively via the MinIO adminstration interface. The example below demonstrates this capability when browsing the `wis2box-incoming` bucket:



CHAPTER

TEN

MONITORING

wis2box has built-in monitoring functions based on Prometheus, Loki and Grafana.

The Grafana endpoint can be visualized at http://localhost/monitoring.

Grafana uses two data sources to display monitoring data:

- Prometheus: actively 'scrapes' data from the configured prometheus-client exporters every X seconds
- Loki: logging endpoint for the Docker containers that compose the wis2box

10.1 Prometheus exporters for wis2box

The exporters for wis2box are based on the Prometheus Python Client

- mqtt_metric_collector: collects data on messages published, using an mqtt-session subscribed to the wis2box-broker

wis2box also analyzes prometheus metrics from MinIO.

Note: For more information see the list of supported MinIO metrics

10.2 Loki logging

The logs of the following Docker containers are sent to Loki:

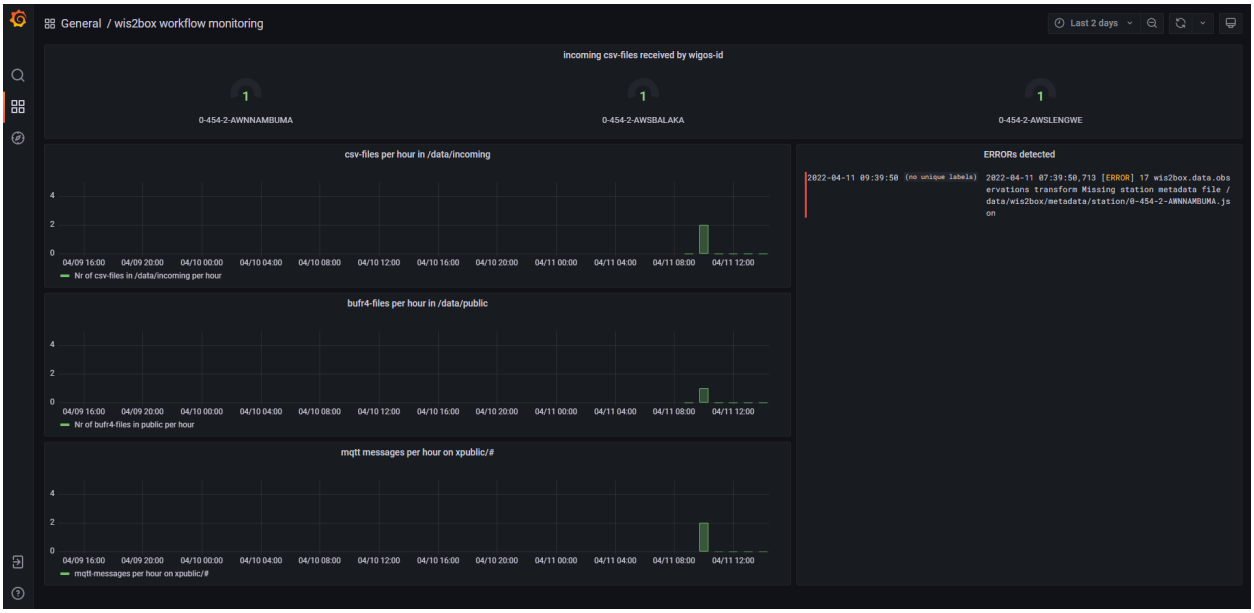
- mqp-publisher
- wis2box
- wis2box-ui
- mosquitto
- wis2box-api
- wis2box-auth

10.3 Monitoring topics

10.3.1 Grafana dashboards

wis2box provides a Grafana dashboard in order to visualize and analyze various metrics.

Go to http://localhost:3000 to see the home dashboard of wis2box once the stack is running.

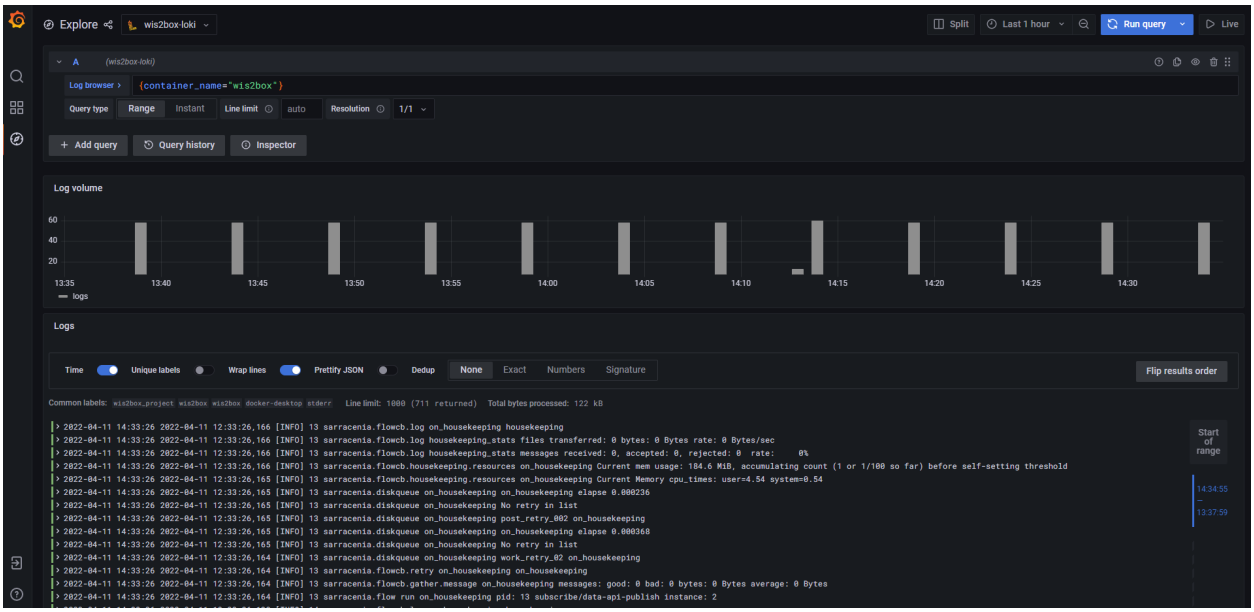


Note: The dashboard configuration can be found in `docker/grafana/dashboards/home.json`.

10.3.2 Exploring logs

You can explore logs by selecting explore from the side-bar in Grafana.

Select `wis2box-loki` as a data source to browse the logs produced by the Docker containers that compose wis2box:


Explore
wis2box-loki
Split
Last 1 hour
Run query
Live
Log browser
{container_name="wis2box"}
Query type
Range
Instant
Line limit
auto
Resolution
1/1
Add query
Query history
Inspector
Log volume
60
40
20
13:35
13:40
13:45
13:50
13:55
14:00
14:05
14:10
14:15
14:20
14:25
14:30
logs
Logs
Time
Unique labels
Wrap lines
Prettify JSON
Dedup
None
Exact
Numbers
Signature
Flip results order
Common labels:
Line limit: 1000 (711 returned)
Total bytes processed: 122 kB
2022-04-11 14:33:26 2022-04-11 12:33:26,166 [INFO] 13 sarracenia.flowcb.log on_housekeeping housekeeping
2022-04-11 14:33:26 2022-04-11 12:33:26,166 [INFO] 13 sarracenia.flowcb.log housekeeping_stats files transferred: 0 bytes: 0 Bytes rate: 0 Bytes/sec
2022-04-11 14:33:26 2022-04-11 12:33:26,166 [INFO] 13 sarracenia.flowcb.log housekeeping_stats messages received: 0, accepted: 0, rejected: 0 rate: 0%
2022-04-11 14:33:26 2022-04-11 12:33:26,166 [INFO] 13 sarracenia.flowcb.housekeeping.resources on_housekeeping Current mem usage: 184.6 MiB, accumulating count (1 or 1/100 so far) before self-setting threshold
2022-04-11 14:33:26 2022-04-11 12:33:26,165 [INFO] 13 sarracenia.flowcb.housekeeping.resources on_housekeeping Current Memory cpu_times: user=4.54 system=0.54
2022-04-11 14:33:26 2022-04-11 12:33:26,165 [INFO] 13 sarracenia.diskqueue on_housekeeping on_housekeeping elapse 0.000236
2022-04-11 14:33:26 2022-04-11 12:33:26,165 [INFO] 13 sarracenia.diskqueue on_housekeeping No retry in list
2022-04-11 14:33:26 2022-04-11 12:33:26,165 [INFO] 13 sarracenia.diskqueue on_housekeeping post_retry_002 on_housekeeping
2022-04-11 14:33:26 2022-04-11 12:33:26,165 [INFO] 13 sarracenia.diskqueue on_housekeeping on_housekeeping elapse 0.000368
2022-04-11 14:33:26 2022-04-11 12:33:26,165 [INFO] 13 sarracenia.diskqueue on_housekeeping No retry in list
2022-04-11 14:33:26 2022-04-11 12:33:26,164 [INFO] 13 sarracenia.diskqueue on_housekeeping work_retry_02 on_housekeeping
2022-04-11 14:33:26 2022-04-11 12:33:26,164 [INFO] 13 sarracenia.flowcb.retry on_housekeeping on_housekeeping
2022-04-11 14:33:26 2022-04-11 12:33:26,164 [INFO] 13 sarracenia.flowcb.gather.message on_housekeeping messages: good: 0 bad: 0 bytes: 0 Bytes average: 0 Bytes
2022-04-11 14:33:26 2022-04-11 12:33:26,164 [INFO] 13 sarracenia.flow run on_housekeeping pid: 13 subscribe/data-api-publish instance: 2
Start of range
14:34:55
13:37:59

CHAPTER

ELEVEN

SERVICES

wis2box provides a number of data access services and mechanisms in providing data to users, applications and beyond.

11.1 Discovery Catalogue

The discovery catalogue is powered by OGC API - Records and is located at http://localhost:8999/oapi/collections/discovery-metadata

The OGC API endpoint is located by default at http://localhost:8999/oapi. The discovery catalogue endpoint is located at http://localhost:8999/oapi/collections/discovery-metadata

Below are some examples of working with the discovery catalogue.

- description of catalogue: http://localhost:8999/oapi/collections/discovery-metadata
- catalogue queryables: http://localhost:8999/oapi/collections/discovery-metadata/queryables
- catalogue queries
  - records (browse): http://localhost:8999/oapi/collections/discovery-metadata/items
  - query by spatial (bounding box): http://localhost:8999/oapi/collections/discovery-metadata/items?bbox=32,-17,36,-8
  - query by temporal extent (since): http://localhost:8999/oapi/collections/discovery-metadata/items?datetime=2021/..
  - query by temporal extent (before): http://localhost:8999/oapi/collections/discovery-metadata/items?datetime=../2022
  - query by freetext: http://localhost:8999/oapi/collections/discovery-metadata/items?q=observations

---

Note:

- adding `f=json` to URLs will provide the equivalent JSON/GeoJSON representations
- query predicates (`datetime`, `bbox`, `q`, etc.) can be combined

See also:

Data access

11.2 Data API

wis2box data is made available via OGC API - Features and is located at http://localhost:8999/oapi standards.

The OGC API endpoint is located by default at http://localhost:8999/oapi

Below are some examples of working with the discovery catalogue.

Note:

- the examples below use the `mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP` collection as described in the *Quickstart*. For other dataset collections, use the same query patterns below, substituting the collection id accordingly

---

- list of dataset collections: http://localhost:8999/oapi/collections
- collection description: http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP
- collection queryables: http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP/queryables
- collection items (browse): http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP/items
- collection queries
  - set limit/offset (paging): http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP/items?limit=1&startindex=2
  - query by spatial (bounding box): http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP/items?bbox=32,-17,36,-8
  - query by temporal extent (since): http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP/items?datetime=2021/..
  - query by temporal extent (before): http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP/items?datetime=../2022

Note:

- adding `f=json` to URLs will provide the equivalent JSON/GeoJSON representations
- query predicates (`datetime`, `bbox`, `q`, etc.) can be combined

See also:

Data access

11.2.1 Management API

The Data API also provides a management API to manage resources in alignment with OGC API - Features - Part 4: Create, Replace, Update and Delete.

11.3 SpatioTemporal Asset Catalog (STAC)

The wis2box SpatioTemporal Asset Catalog (STAC) endpoint can be found at:

http://localhost:8999/stac

…providing the user with a crawlable catalogue of all data on a wis2box.

11.4 Web Accessible Folder (WAF)

The wis2box Web Accessible Folder publich bucket endpoint can be found at:

http://localhost:8999/data/

…providing the user with a crawlable online folder of all data on a wis2box.

11.5 Broker

The wis2box broker is powered by MQTT and can be found at:

mqtt://localhost:1883

…providing a PubSub capability for event driven subscription and access.

11.6 Adding services

wis2box's architecture allows for additional services as required by adding Docker containers. Examples of additional services include adding a container for a samba share or FTP server. Key considerations for adding services:

- Storage buckets can be found at http://minio:9000
- Elasticsearch indexes can be found at the container/URL `http://elasticsearch:9200`

Examples of additional services can be found in `docker/extras`.

CHAPTER

TWELVE

AUTHENTICATION AND ACCESS CONTROL

wis2box provides built in access control for the WAF and API on a topic hierarchy basis. Configuration is done using the wis2box command line utility. Authentication tokens are only required for topics that have access control configured.

12.1 Adding Access Control

All topic hierarchies in wis2box are open by default. A topic becomes closed, with access control applied, the first time a token is generated for a topic hierarchy.

Note: Make sure you are logged into the wis2box container when using the wis2box CLI

```
wis2box auth add-token --topic-hierarchy mwi.mwi_met_centre.data.core.weather.surface-
↪based-observations.SYNOP mytoken
```

If no token is provided, a random string will be generated. Be sure to the record token now, there is no way to retrieve it once it is lost.

12.2 Authenticating

Token credentials can be validated using the wis2box command line utility.

```
wis2box auth show
wis2box auth has-access --topic-hierarchy mwi.mwi_met_centre.data.core.weather.surface-
↪based-observations.SYNOP mytoken
wis2box auth has-access --topic-hierarchy mwi.mwi_met_centre.data.core.weather.surface-
↪based-observations.SYNOP notmytoken
```

Once a token has been generated, access to any data of that topic in the WAF or API requires token authentication. Tokens are passed as a bearer token in the Authentication header or as an argument appended to the URI. Headers can be easily added to requests using cURL.

```
curl -H "Authorization: Bearer mytoken" "http://localhost:8999/oapi/collections/mwi.mwi_
↪met_centre.data.core.weather.surface-based-observations.SYNOP"
curl -H "Authorization: Bearer notmytoken" "http://localhost:8999/oapi/collections/mwi.
↪mwi_met_centre.data.core.weather.surface-based-observations.SYNOP"
```

12.3 Removing Access Control

A topic becomes open and no longer requires authentication when all tokens have been deleted. This can be done by deleting individual tokens, or all tokens for a given topic hierarchy.

```
wis2box auth remove-tokens --topic-hierarchy mwi.mwi_met_centre.data.core.weather.
↪surface-based-observations.SYNOP
wis2box auth show
```

12.4 Extending Access Control

wis2box provides access control out of the box with subrequests to wis2box-auth. wis2box-auth could be replaced in nginx for another auth server like Gluu or a Web SSO like LemonLDAP or Keycloak. These services are not yet configurable via the wis2box command line utility.

wis2box is intentionally plug and playable. Beyond custom authentication servers, extending wis2box provides an overview of more modifications that can be made to wis2box.

CHAPTER

THIRTEEN

DATA ACCESS

13.1 Overview

This section provides examples of interacting with wis2box data services as described in *Services* using a number of common tools and software packages.

13.2 API

13.2.1 Using Python, requests and Pandas

Python is a popular programming language which is heavily used in the data science domains. Python provides high level functionality supporting rapid application development with a large ecosystem of packages to work with weather/climate/water data.

Let's use the Python requests package to further interact with the wis2box API, and Pandas to run some simple summary statistics.

[1]:
```
import json

import requests

def pretty_print(input):
    print(json.dumps(input, indent=2))


# define the endpoint of the OGC API
api = 'http://localhost:8999/oapi'
```

Stations

Let's find all the stations in our wis2box:

[2]:
```
url = f'{api}/collections/stations/items?limit=50'

response = requests.get(url).json()

print(f"Number of stations: {response['numberMatched']}")
```

(continues on next page)

(continued from previous page)

```
print('Stations:\n')
for station in response['features']:
    print(station['properties']['name'])
```

```
Number of stations: 26
Stations:

NAMBUMA
BALAKA
BILIRA
CHIDOOLE
CHIKANGAWA
CHIKWEO
CHINGALE
KALAMBO
KASIYA AWS
KASUNGU NATIONAL PARK AWS
KAWALAZI
KAYEREKERA
LENGWE NATIONAL PARK
LOBI AWS
MAKANJIRA
MALOMO
MISUKU
MLARE
MLOMBA
MTOSA BENGA
NAMITAMBO
NANKUMBA
NKHOMA UNIVERSITY
NKHULAMBE
NYACHILENDA
TOLEZA
```

Discovery Metadata

Now, let's find all the dataset that are provided by the above stations. Each dataset is identified by a WIS 2.0 discovery metadata record.

[3]:
```
url = f'{api}/collections/discovery-metadata/items'

response = requests.get(url).json()

print('Datasets:\n')
for dataset in response['features']:
    print(f"id: {dataset['properties']['id']}, title: {dataset['properties']['title']}")
```

```
Datasets:

id: data.core.test-passthrough, title: Surface weather observations (passthrough)
id: mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP, title:␣
↪Surface weather observations (hourly)
```

Let's find all the data access links associated with the Surface weather observations (hourly) dataset:

[4]:
```
dataset_id = 'mwi.mwi_met_centre.data.core.weather.surface-based-observations.SYNOP'

url = f"{api}/collections/discovery-metadata/items/{dataset_id}"

response = requests.get(url).json()

print('Data access links:\n')
for link in response['links']:
    print(f"{link} {link['href']} ({link['type']}) {link['rel']}")
    link['rel']

[link['href'] for link in response['links']]
```

```
Data access links:

{'rel': 'self', 'type': 'application/geo+json', 'title': 'This document as GeoJSON',
↪'href': 'http://localhost:8999/oapi/collections/discovery-metadata/items/mwi.mwi_met_
↪centre.data.core.weather.surface-based-observations.SYNOP?f=json'} http://localhost:
↪8999/oapi/collections/discovery-metadata/items/mwi.mwi_met_centre.data.core.weather.
↪surface-based-observations.SYNOP?f=json (application/geo+json) self
{'rel': 'alternate', 'type': 'application/ld+json', 'title': 'This document as RDF (JSON-
↪LD)', 'href': 'http://localhost:8999/oapi/collections/discovery-metadata/items/mwi.mwi_
↪met_centre.data.core.weather.surface-based-observations.SYNOP?f=jsonld'} http://
↪localhost:8999/oapi/collections/discovery-metadata/items/mwi.mwi_met_centre.data.core.
↪weather.surface-based-observations.SYNOP?f=jsonld (application/ld+json) alternate
{'rel': 'alternate', 'type': 'text/html', 'title': 'This document as HTML', 'href':
↪'http://localhost:8999/oapi/collections/discovery-metadata/items/mwi.mwi_met_centre.
↪data.core.weather.surface-based-observations.SYNOP?f=html'} http://localhost:8999/oapi/
↪collections/discovery-metadata/items/mwi.mwi_met_centre.data.core.weather.surface-
↪based-observations.SYNOP?f=html (text/html) alternate
{'rel': 'collection', 'type': 'application/json', 'title': 'Discovery metadata', 'href':
↪'http://localhost:8999/oapi/collections/discovery-metadata'} http://localhost:8999/
↪oapi/collections/discovery-metadata (application/json) collection
```

[4]:
```
['http://localhost:8999/oapi/collections/discovery-metadata/items/mwi.mwi_met_centre.
↪data.core.weather.surface-based-observations.SYNOP?f=json',
 'http://localhost:8999/oapi/collections/discovery-metadata/items/mwi.mwi_met_centre.
↪data.core.weather.surface-based-observations.SYNOP?f=jsonld',
 'http://localhost:8999/oapi/collections/discovery-metadata/items/mwi.mwi_met_centre.
↪data.core.weather.surface-based-observations.SYNOP?f=html',
 'http://localhost:8999/oapi/collections/discovery-metadata']
```

Let's use the OGC API - Features (OAFeat) link to drill into the observations for Chidoole station

[5]:
```
dataset_api_link = 'http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.
↪weather.surface-based-observations.SYNOP'

dataset_api_link
```

[5]:
```
'http://localhost:8999/oapi/collections/mwi.mwi_met_centre.data.core.weather.surface-
↪based-observations.SYNOP'
```

Observations

Let's inspect some of the data in the API's raw GeoJSON format:

[6]:
```
url = f'{dataset_api_link}/items'

query_parameters = {
    'wigos_station_identifier': '0-454-2-AWSCHIDOOLE',
    'limit': 10000,
    'name': 'air_temperature'
}

response = requests.get(url, params=query_parameters).json()

pretty_print(response['features'][0])
```


```
{
  "id": "WIGOS_0-454-2-AWSCHINGALE_20220112T135500-25",
  "reportId": "WIGOS_0-454-2-AWSCHINGALE_20220112T135500",
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [
      35.11,
      -15.24,
      623.0
    ]
  },
  "properties": {
    "wigos_station_identifier": "0-454-2-AWSCHINGALE",
    "phenomenonTime": "2022-01-12T13:55:00Z",
    "resultTime": "2022-01-12T13:55:00Z",
    "name": "air_temperature",
    "value": 24.85,
    "units": "Celsius",
    "description": null,
    "metadata": [
      {
        "name": "station_or_site_name",
        "value": null,
        "units": "CCITT IA5",
        "description": "Chingale"
      },
      {
        "name": "station_type",
        "value": 0,
        "units": "CODE TABLE",
        "description": "Automatic"
      },
      {
        "name": "height_of_barometer_above_mean_sea_level",
        "value": 624.0,
        "units": "m",
        "description": null
```


(continues on next page)

(continued from previous page)


```
      },
      {
        "name": "height_of_sensor_above_local_ground_or_deck_of_marine_platform",
        "value": 1.5,
        "units": "m",
        "description": null
      }
    ],
    "index": 25,
    "fxxyyy": "012101",
    "id": "WIGOS_0-454-2-AWSCHINGALE_20220112T135500-25"
  }
}
```


Let's inspect what's measured at Chidoole:

[7]:
```
print('Observed property:\n')
feature = response['features'][9]
print(f"{feature['properties']['name']} ({feature['properties']['units']})")
```

```
Observed property:

air_temperature (Celsius)
```

Pandas

Let's use the GeoJSON to build a more user-friendly table

[8]:
```
import pandas as pd

datestamp = [obs['properties']['resultTime'] for obs in response['features']]
air_temperature = [obs['properties']['value'] for obs in response['features']]

d = {
    'Date/Time': datestamp,
    'Air temperature (°C)': air_temperature
}

df = pd.DataFrame(data=d)
```

[9]:
```
df
```

[9]:

	Date/Time	Air temperature (°C)
0	2022-01-12T13:55:00Z	24.85
1	2022-01-12T14:55:00Z	27.25
2	2022-01-12T15:55:00Z	26.65
3	2022-01-12T16:55:00Z	25.95
4	2022-01-12T17:55:00Z	25.45
...	...	...
5101	2022-06-09T12:55:00Z	21.35
5102	2022-06-09T13:55:00Z	22.25
5103	2022-06-09T14:55:00Z	20.25

(continues on next page)

(continued from previous page)

```
5104  2022-06-10T12:55:00Z                 23.75
5105  2022-06-10T14:55:00Z                 21.15

[5106 rows x 2 columns]
```

[10]:
```
print("Time extent\n")
print(f'Begin: {df["Date/Time"].min()}')
print(f'End: {df["Date/Time"].max()}')

print("Summary statistics:\n")
df[['Air temperature (°C)']].describe()
```

```
Time extent

Begin: 2022-01-12T13:55:00Z
End: 2022-06-10T14:55:00Z
Summary statistics:
```

[10]:
```
       Air temperature (°C)
count           5106.000000
mean              23.541559
std                4.053172
min               13.550000
25%               20.950000
50%               23.350000
75%               26.350000
max               37.850000
```

[ ]:
```

```

13.2.2 Using Python and OWSLib

OWSLib is a Python package which provides Pythonic access to OGC APIs and web services. Let's see how easy it is to work with wis2box with standards-based tooling:

[1]:
```
from owslib.ogcapi.features import Features

import pandas as pd

def pretty_print(input):
    print(json.dumps(input, indent=2))


api = 'http://localhost:8999/oapi'
```

Let's load the wis2box API into OWSLib and inspect some data

[2]:
```
oafeat = Features(api)

collections = oafeat.collections()
```

(continues on next page)

(continued from previous page)

```
print(f'This OGC API Features endpoint has {len(collections["collections"])} datasets')

for dataset in collections['collections']:
    print(dataset['title'])

malawi_obs = oafeat.collection_items('mwi.mwi_met_centre.data.core.weather.surface-based-
↪observations.SYNOP')
malawi_obs_df = pd.DataFrame(malawi_obs['features'])

# then filter by station
obs = oafeat.collection_items('mwi.mwi_met_centre.data.core.weather.surface-based-
↪observations.SYNOP', wigos_station_identifier='0-454-2-AWSCHIDOOLE', name='air_
↪temperature', limit=10000)

datestamp = [obs['properties']['resultTime'] for obs in obs['features']]
air_temperature = [obs['properties']['value'] for obs in obs['features']]

d = {
    'Date/Time': datestamp,
    'Air temperature (°C)': air_temperature
}

df = pd.DataFrame(data=d)
```

```
This OGC API Features endpoint has 4 datasets
Surface weather observations (passthrough)
Discovery metadata
Stations
Surface weather observations (hourly)
```

[3]:
```
df.dtypes
```

[3]:
```
Date/Time                object
Air temperature (°C)    float64
dtype: object
```

[4]:
```
df.head(3)
```

[4]:
```
              Date/Time  Air temperature (°C)
0  2022-01-12T13:55:00Z                 24.85
1  2022-01-12T14:55:00Z                 27.25
2  2022-01-12T15:55:00Z                 26.65
```

[5]:
```
print("Time extent\n")
print(f'Begin: {df["Date/Time"].min()}')
print(f'End: {df["Date/Time"].max()}')

print("Summary statistics:\n")
df[['Air temperature (°C)']].describe()
```

```
Time extent

Begin: 2022-01-12T13:55:00Z
```

(continues on next page)

(continued from previous page)

```
End: 2022-06-10T14:55:00Z
Summary statistics:
```

[5]:
```
       Air temperature (°C)
count           5106.000000
mean              23.541559
std                4.053172
min               13.550000
25%               20.950000
50%               23.350000
75%               26.350000
max               37.850000
```

[ ]:
```
```

13.2.3 R

R is a common programming language for data analysis and visualization. R provides easy access to various statiscal analysis libraries. We are going to use the R libraries: sf to load features, dplyr for data manipulation, and

Install Requirements

[ ]:
```
install.packages("sf")
install.packages("dplyr")
```

Import Requirements

[1]:
```
library(sf)
library(dplyr)

oapi <- "http://oapi/oapi" # jupyter is run through docker
#oapi = http://localhost:8999/oapi # jupyter is run on host machine
```

```
Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 8.2.1; sf_use_s2() is TRUE


Attaching package: ‘dplyr’


The following objects are masked from ‘package:stats’:

    filter, lag


The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Stations

[2]:
```
stations <- read_sf(paste0(oapi,"/collections/stations/items?f=json"))
print(stations)
```

```
Simple feature collection with 7 features and 5 fields
Geometry type: POINT
Dimension:     XYZ
Bounding box:  xmin: 33.67305 ymin: -15.84052 xmax: 35.27428 ymax: -9.92951
z_range:       zmin: 618 zmax: 1288
Geodetic CRS:  WGS 84
# A tibble: 7 × 6
  wigos_station_identifier              name        url   status    id                ␣
↪geometry
  <chr>                 <chr>       <chr> <chr>  <int>               <POINT [°]>
1 0-454-2-AWSLOBI       LOBI AWS    http... opera... 65618 Z (34.07244 -14.39528 12...
2 0-454-2-AWSKAYEREKERA KAYEREKERA  http... opera... 91840 Z (33.67305 -9.92951 848)
3 0-454-2-AWSMALOMO     MALOMO      http... opera... 91873 Z (33.83727 -13.14202 10...
4 0-454-2-AWSNKHOMA     NKHOMA UNI... http... opera... 91875 Z (34.10468 -14.04422 12...
5 0-454-2-AWSTOLEZA     TOLEZA      http... opera... 91880    Z (34.955 -14.948 764)
6 0-454-2-AWSNAMITAMBO  NAMITAMBO   http... opera... 91885 Z (35.27428 -15.84052 80...
7 0-454-2-AWSBALAKA     BALAKA      http... opera... 91893 Z (34.96667 -14.98333 61...
```

Discovery Metadata

[3]:
```
discovery_metadata <- read_sf(paste0(oapi,"/collections/discovery-metadata/items"))
print(discovery_metadata)
```

```
Simple feature collection with 1 feature and 13 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 32.68817 ymin: -16.8013 xmax: 35.7719 ymax: -9.230599
Geodetic CRS:  WGS 84
# A tibble: 1 × 14
  identifier externalId title description themes providers language type  extent
  <chr>      <chr>      <chr> <chr>       <chr>  <chr>     <chr>    <chr> <chr>
1 data.core... "[ { \"sc... Surf... Surface we... "[ { ... "[ { \"n... en       data... "
↪{ \"...
# ... with 5 more variables: created <date>, rights <chr>,
#   X_metadata.anytext <chr>, id <chr>, geometry <POLYGON [°]>
```

Observations

[4]:
```
malawi_obs <- read_sf(paste0(oapi,"/collections/mwi.mwi_met_centre.data.core.weather.
↪surface-based-observations.SYNOP/items"))
print(malawi_obs)
```

```
Simple feature collection with 10 features and 7 fields
Geometry type: POINT
Dimension:     XYZ
Bounding box:  xmin: 35.27 ymin: -15.84 xmax: 35.27 ymax: -15.84
```

(continues on next page)

(continued from previous page)

```
z_range:       zmin: 806 zmax: 806
Geodetic CRS:  WGS 84
# A tibble: 10 × 8
   identifier  phenomenonTime      resultTime          wigos_station_i... metadata
   <chr>       <dttm>              <dttm>              <chr>              <chr>
 1 WIGOS_0-45... 2021-07-07 14:55:00 2022-02-21 14:15:14 0-454-2-AWSNAMI... "[ { \"...
 2 WIGOS_0-45... 2021-07-07 15:55:00 2022-02-21 14:15:14 0-454-2-AWSNAMI... "[ { \"...
 3 WIGOS_0-45... 2021-07-07 16:55:00 2022-02-21 14:15:14 0-454-2-AWSNAMI... "[ { \"...
 4 WIGOS_0-45... 2021-07-07 17:55:00 2022-02-21 14:15:14 0-454-2-AWSNAMI... "[ { \"...
 5 WIGOS_0-45... 2021-07-07 18:55:00 2022-02-21 14:15:14 0-454-2-AWSNAMI... "[ { \"...
 6 WIGOS_0-45... 2021-07-07 19:55:00 2022-02-21 14:15:15 0-454-2-AWSNAMI... "[ { \"...
 7 WIGOS_0-45... 2021-07-07 20:55:00 2022-02-21 14:15:15 0-454-2-AWSNAMI... "[ { \"...
 8 WIGOS_0-45... 2021-07-07 21:55:00 2022-02-21 14:15:15 0-454-2-AWSNAMI... "[ { \"...
 9 WIGOS_0-45... 2021-07-07 22:55:00 2022-02-21 14:15:15 0-454-2-AWSNAMI... "[ { \"...
10 WIGOS_0-45... 2021-07-07 23:55:00 2022-02-21 14:15:15 0-454-2-AWSNAMI... "[ { \"...
# ... with 3 more variables: observations <chr>, id <chr>, geometry <POINT [°]>
```

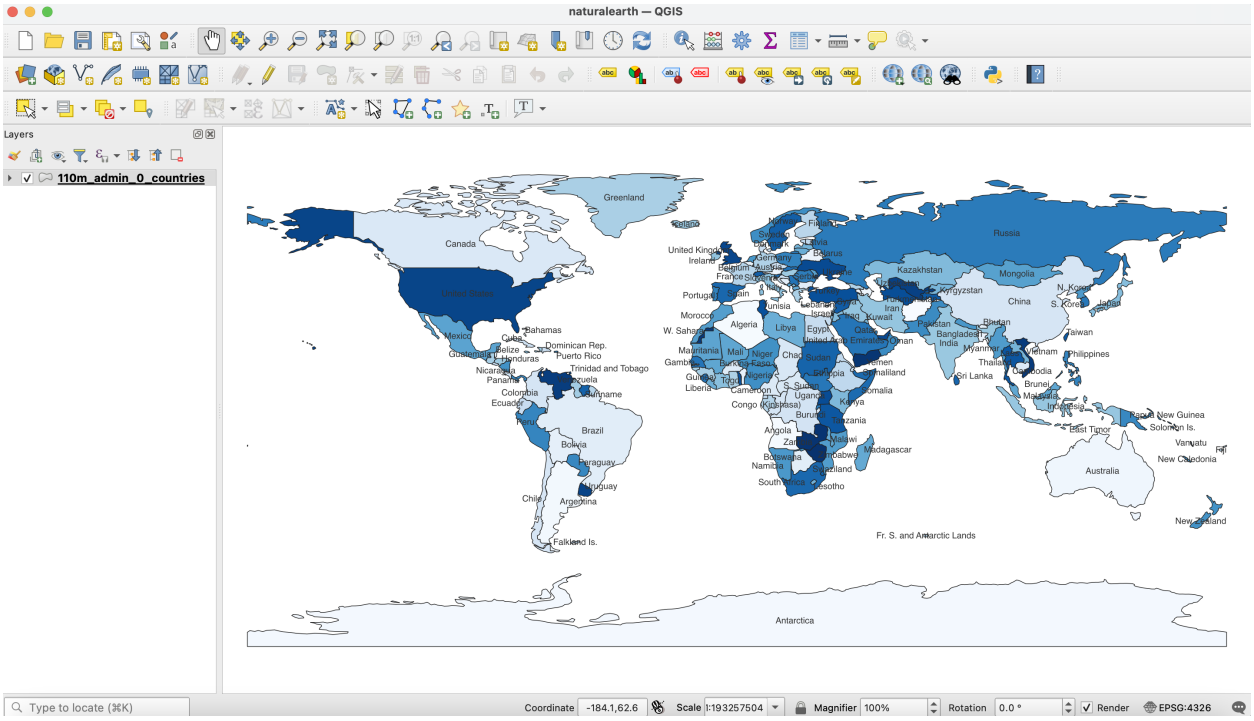
[ ]:

13.2.4 Using QGIS

Overview

This section provides examples of interacting with wis2box API using QGIS.

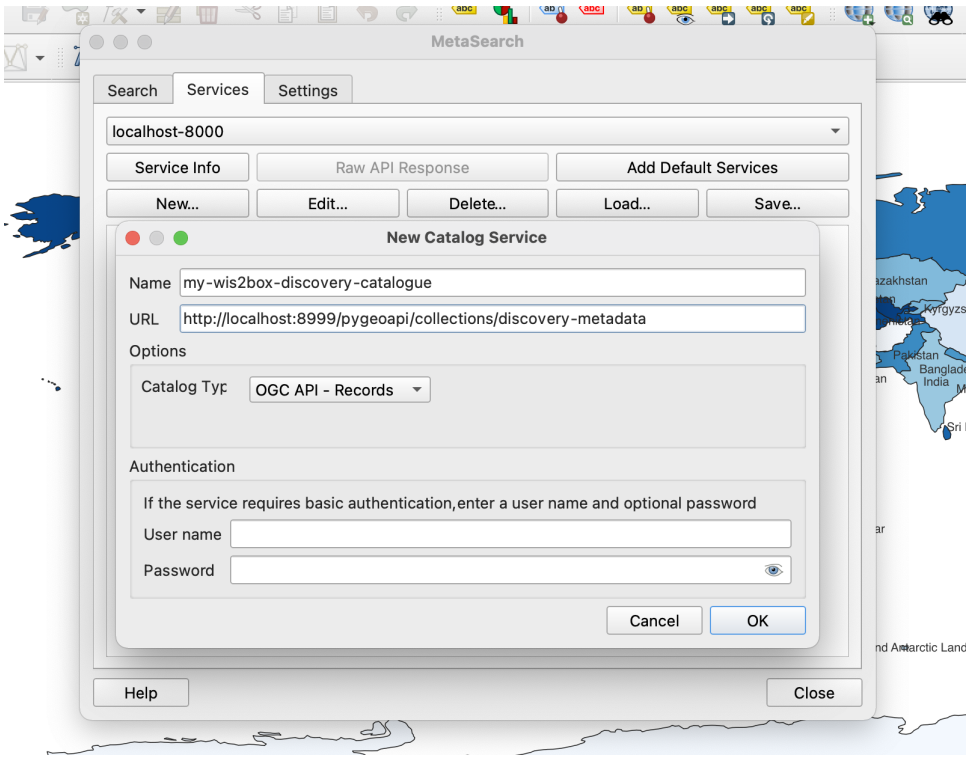
QGIS is a free and open-source cross-platform desktop GIS application that supports viewing, editing, and analysis of geospatial data. QGIS supports numerous format and encoding standards, which enables plug-and-play interoperability with wis2box data and discovery metadata.



Accessing the discovery catalogue

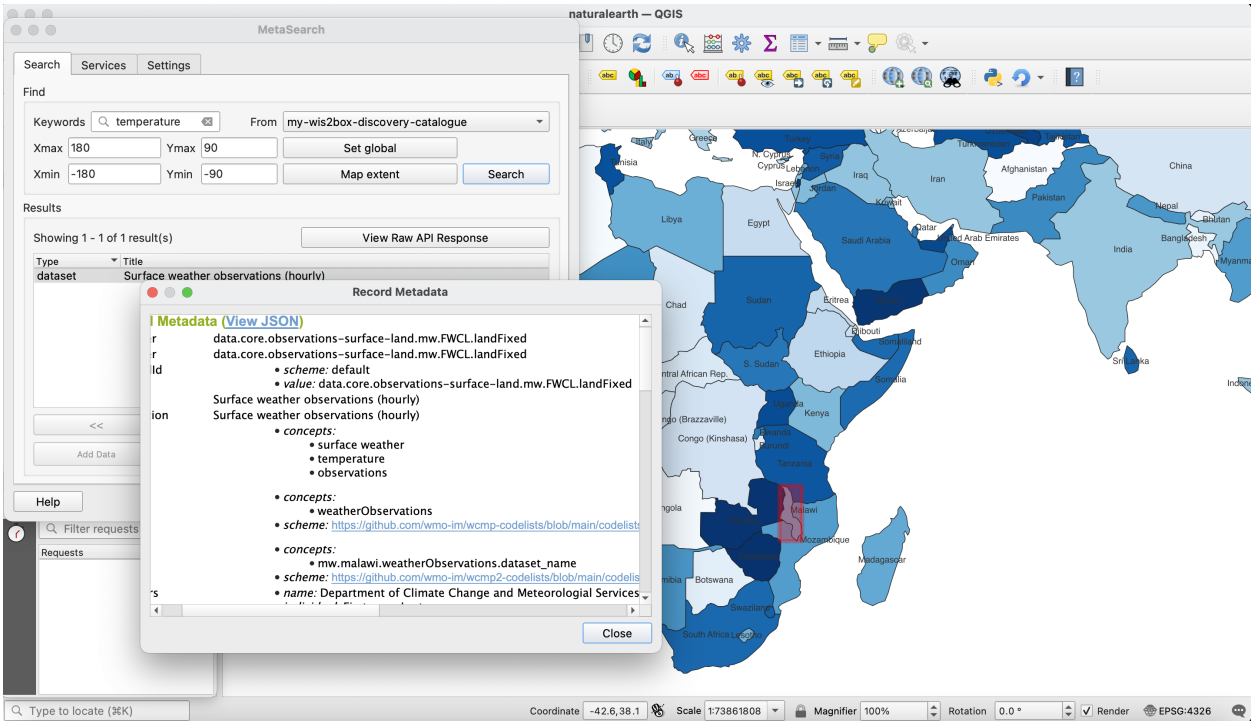
QGIS provides support for the OGC API - Records standard (discovery). To interact with the wis2box discovery catalogue:

- from the QGIS menu, select *Web -> MetaSearch -> MetaSearch*
- click the "Services" tab
- click "New"
- enter a name for the discovery catalogue endpoint
- enter the URL to the discovery catalogue endpoint (i.e. `http://localhost:8999/oapi/collections/discovery-metadata`)
- ensure "Catalogue Type" is set to "OGC API - Records"
- click "OK"



This adds the discovery catalogue to the MetaSearch catalogue registry. Click "Service Info" to display the properties of the discovery catalogue service metadata.

To search the discovery catalogue, click the "Search" tab, which will provide the ability to search for metadata records by bounding box and/or full text search. Click the "Search" button to search the discovery catalogue and visualize search results. Clicking on metadata records in the search result table will show footprints on the map to help provide the location of the search result. Double-clicking a search result will show the entire metadata record.

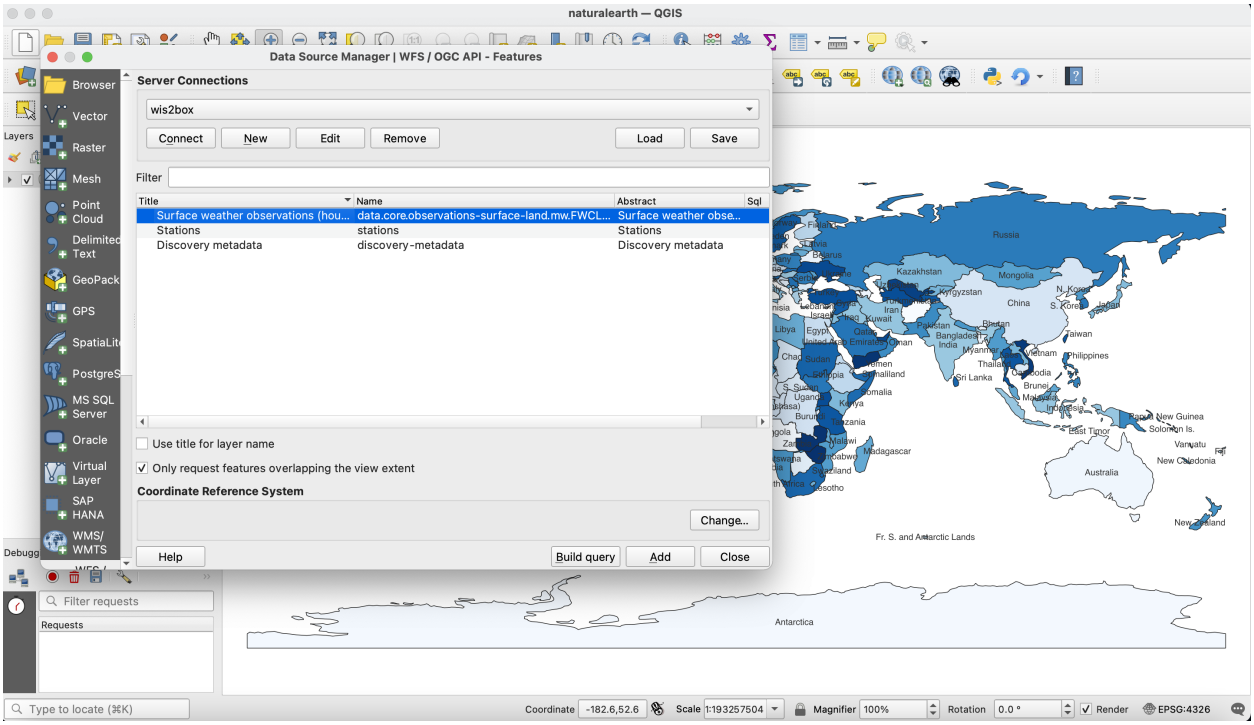


Note: For more information on working with catalogues, consult the official QGIS MetaSearch documentation.

Visualizing stations

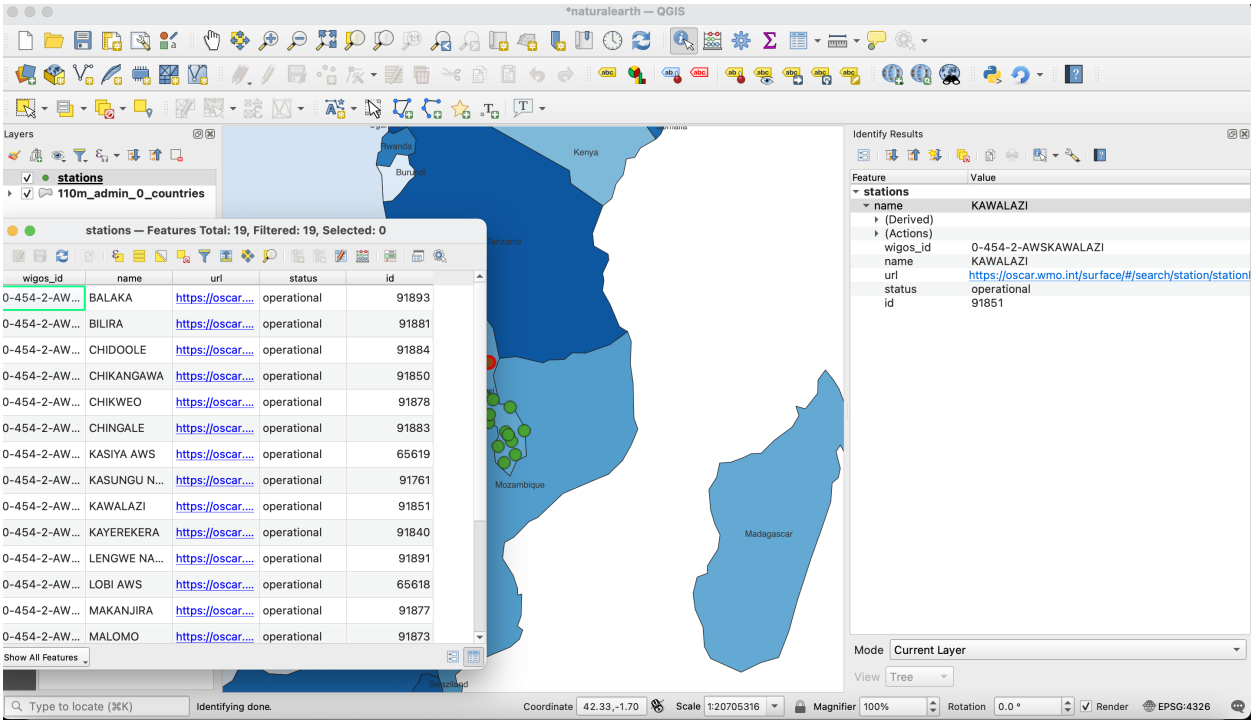
QGIS provides support for the OGC API - Features standard (access). To interact with the wis2box API:

- from the QGIS menu, select *Layer -> Add Layer -> Add WFS Layer...*
- click "New"
- enter a name for the API endpoint
- enter the URL to the API endpoint (i.e. `http://localhost:8999/oapi`)
- under "WFS Options", set "Version" to "OGC API - Features"
- click "OK"
- click "Connect"



A list of collections is displayed. Select the "Stations" collection and click "Add". The Stations collection is now added to the map. To further explore:

- click on the "Identify" (i) and click on a station to display station properties
- select *Layer -> Open Attribute Table* to open all stations in a tabular view



Note that the same QGIS workflow can be executed for any other collection listed from wis2box API.

Note: For more information on working with OGC API - Features, consult the official QGIS WFS documentation.

Summary

The above examples provide a number of ways to utilize the wis2box API from the QGIS desktop GIS application.

13.3 PubSub

13.3.1 Downloading data from WIS2 using pywis-pubsub

Overview

This section provides examples of using the pywis-pubsub tool to download data from WIS2 global services. WIS2 global services include a Global Broker that provides users the ability to subscribe to data (via topics) and download to their local environment / workstation / decision support system from the WIS2 Global Cache.

The pywis-pubsub tool

This repository includes docker files to help you subscribe and download data from the WIS2 network, by using `pywis-pubsub` inside the wis2box-subscribe-download container.

`pywis-pubsub` is a Python package that provides publish, subscription and download capability of data from WIS2 infrastructure services.

Before starting the wis2box-subscribe-download container, the default configuration (provided in `docker/wis2box-subscribe-download/local.yml`) must be updated, by defining the URL of the MQTT broker as well as the desired topics to subscribe to.

In addition, the storage path should be updated to specify where the downloaded should be saved to.

```
# fully qualified URL of broker
broker: mqtts://username:password@host:port

# whether to run checksum verification when downloading data (default true)
verify_data: true

# whether to validate broker messages (default true)
validate_message: true

# list of 1..n topics to subscribe to
topics:
    - 'cache/a/wis2/topic1/#'
    - 'cache/a/wis2/topic2/#'

# storage: filesystem
storage:
    type: fs
    options:
        path: /tmp/foo/bar
```

To start a continuous subscribe-and-download process, run the `wis2box-subscribe-download` container as follows (`-d` for detached mode, `--build` to ensure changes in local.yml are built into the container):

```
docker-compose -f docker/docker.subscribe-download.yml up -d --build
```

To stop the subscribe-and-download process, run the following command:

```
docker-compose -f docker/docker.subscribe-download.yml down
```

Running pywis-pubsub interactively

pywis-pubsub can also be run interactively from inside the wis2box main container as follows:

```
# login to wis2box main container
python3 wis2box-ctl.py login

# edit a local configuration by using docker/wis2box-subscribe-download/local.yml as a␣
↪template
vi /data/wis2box/local.yml

# connect, and simply display data notifications
pywis-pubsub subscribe --config local.yml

# connect, and additionally download messages
pywis-pubsub subscribe --config local.yml --download

# connect, and filter messages by bounding box geometry
pywis-pubsub subscribe --config local.yml --bbox=-142,42,-52,84
```

Summary

The above examples provide examples of using pywis-pubsub to subscribe and download data from WIS2 global services.

13.3.2 Using Python and paho-mqtt

This example will use widely available and used python language and libraries to download some announcements, and then retrieve the corresponding data, using only the paho-mqtt client library, in addition to Python standard libraries.

[1]:
```
import json
import paho.mqtt.client as mqtt
import random
import urllib
import urllib.request


host='localhost'
user='wis2box'
password='wis2box'

r = random.Random()
```

(continues on next page)

(continued from previous page)

```
clientId='MyQueueName'+ f"{r.randint(1,1000):04d}"
# number of messages to subscribe to.
messageCount = 0
messageCountMaximum = 5

# maximum size of data download to print.
sizeMaximumThreshold = 1023
```

The above imports the required modules. It is also assumed that `localhost` is set up and is publishing messages. Message queueing protocols provide real-time notification about availability of products.

The standard Python package used to subscribe to messages is `paho-mqtt` (`paho.mqtt.client`). The package uses callbacks.

Note that `messageCount` is used to limit the length of the demonstration (otherwise infinite, as it is a continuous flow).

Let's investigate our callbacks.

[2]:
```
def sub_connect(client, userdata, flags, rc, properties=None):
    print("on connection to subscribe: ", mqtt.connack_string(rc))
    for s in ["origin/#"]:
        client.subscribe(s, qos=1)
```

The `sub_connect` callback needed is called when the connection is established, which required to subscribe to topics we are interested in (topics are: `origin/#`, where `/` is a topic separator and `#` is a wildcard for any tree of topics.

The `qos=1` refers to Quality of Service, where 1 establishes reception of messages at least once. `qos=1` is recommended.

The next callback is called every time a message is received, and decodes and prints the message.

To keep the output short for the demonstration, we limit the subscriber to a few messages.

[3]:
```
def sub_message(client, userdata, msg):
    """
    print messages received.  Exit on count received.
    """

    global messageCount,messageCountMaximum

    m = json.loads(msg.payload.decode('utf-8'))

    print(f"message {messageCount} topic: {msg.topic} received: {m}")
    print(f"message {messageCount} data: {getData(m)}")

    messageCount += 1

    if messageCount > messageCountMaximum:
        client.disconnect()
        client.loop_stop()
```

The message handler above calls the `getData()` (below). The messages themselves are usually announcements of data availability, but when data is small, they can include the data itself (inline) in the `content` field. Usually the message refers to the data using a link. Here is a routine to obtain the data given an announcement message:

[4]:
```
def getData(m, sizeMaximum=1000):
    """
```

(continues on next page)

(continued from previous page)

```
    given a message, return the data it refers to
    """

    if 'size' in m and m['size'] > sizeMaximum:
        return f" data too large {m['size']} bytes"
    elif 'content' in m:
        if m['content']['encoding'] == 'base64':
            return b64decode(m['content']['value'])
        else:
            return m['content']['value'].encode('utf-8')
    else:
        url = m['baseUrl'] + '/' + m['relPath']
        with urllib.request.urlopen(url) as response:
            return response.read()
```

The calling code then registers the callbacks, connects to the broker, and starts the event loop:

[ ]:
```
client = mqtt.Client(client_id=clientId, protocol=mqtt.MQTTv5)
client.on_connect = sub_connect
client.on_message = sub_message
client.username_pw_set(user, password)
client.connect(host)

client.loop_forever()
```

```
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
```

(continues on next page)

(continued from previous page)

```
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
on connection to subscribe:  Connection Accepted.
```

[ ]:

13.4 Running the examples

To be able to run these examples, one needs to start up a Jupyter Notebook environment. Below is an example of starting a Jupyter session:

```
git clone https://github.com/wmo-im/wis2box.git
cd docs/source/data-access
jupyter notebook --ip=0.0.0.0 --port=8888
```

When Jupyter starts up it may open a browser window for you. If not you would need to to point a browser at http://localhost:8888 to see the menu of notebooks available in this directory.

13.5 Summary

The above examples provide a number of ways to utilize the wis2box suite of services.

CHAPTER

FOURTEEN

EXTENDING WIS2BOX

At its core, wis2box is a plugin architecture orchestrating all the required components of a node in the WIS 2.0 network. Driven by topic hierarchies, wis2box can be used to process and publish any type of geospatial data beyond the requirements of the WIS 2.0 itself.

In this section we will to explore how wis2box can be extended. wis2box plugin development requires knowledge of how to program in Python as well as Python's packaging and module system.

14.1 Building your own data plugin

The heart of a wis2box data plugin is driven from the `wis2box.data.base` abstract base class (ABC) located in `wis2box/data/base.py`. Any wis2box plugin needs to inherit from `wis2box.data.base.BaseAbstractData`. A minimal example can be found below:

```
from datetime import datetime
from wis2box.data.base import BaseAbstractData

class MyCoolData(BaseAbstractData):
    """Observation data"""
    def __init__(self, defs: dict) -> None:
        super().__init__(defs)

    def transform(self, input_data: Path) -> bool:
        # transform data
        # populate self.output_data with a dict as per:
        self.output_data = {
            'c123': {
                '_meta': {
                    'identifier': 'c123',
                    'relative_filepath': '/path/to/item/',
                    'data_date': datetime_object
                },
                'bufr4': bytes(12356),
                'geojson': geojson_string
            }
        }
        return True
```

The key function that plugin needs to implement is the `transform` function. This function should return a `True` or `False` of the result of the processing, as well as populate the `output_data` property.

The `output_data` property is a `dict` of keys/values. Each key should be the identifier of the item, with the following values `dict`:

The `_meta` element can include the following:

- `identifier`: identifier for report (WIGOS_<WSI>_<ISO8601>)
- `relative_filepath`: path to data, required to publish data with `BaseAbstractData.publish`
- `geometry`: GeoJSON geometry object, required to send geometry with WIS2.0 notification
- `md5`: md5 checksum of encoded data
- `wigos_station_identifier`: WIGOS identifier
- `data_date`: (as Python datetime objects) based on the observed datetime
- `originating_centre`: Originating centre (see Common code table C11)
- `data_category`: Category of data, see BUFR Table A

- `<format-extension>`: 1..n properties for each format representation, with the key being the filename extension. The value of this property can be a string or bytes, depending on whether the underlying data is ASCII or binary, for example

14.2 Packaging

The next step is assembling your plugin using standard Python packaging. All plugin code and configuration files should be made part of the package so that it can operate independently when running in wis2box. For distribution and installation, you have the following options:

- publish to the Python Package Index (PyPI) and install in the wis2node container with `pip3 install wis2box-mypackage`
- `git clone` or download your package, and install via `python3 setup.py install`

See the Python packaging tutorial or Cookiecutter PyPackage for guidance and templates/examples.

Note: It is recommended to name your wis2box packages with the convention `wis2box-MYPLUGIN-NAME`, as well as adding the keywords/topics `wis2box` and `plugin` to help discovery on platforms such as GitHub.

14.3 Integration

Once your package is installed on the wis2box container, the data mappings need to be updated to connect your plugin to a topic hierarchy. See *Data mappings* for more information.

An example plugin for proof of concept can be found in https://github.com/wmo-cop/wis2box-csv-observations

14.4 Example plugins

The following plugins provide useful examples of wis2box plugins implemented by downstream applications.

Plugin(s)	Organization/Project	Description
wis2box-csv-observations	WMO	plugin for CSV surface observation data
wis2box-pyopencdms-plugin	OpenCDMS	plugin for connecting the Open Climate Data Management System to wis2box

CHAPTER

FIFTEEN

DEVELOPMENT

wis2box is developed as a free and open source project on GitHub. The wis2box codebase can be found at https://github.com/wmo-im/wis2box.

15.1 Testing

wis2box continuous integration (CI) testing is managed by GitHub Actions. All commits and pull requests to wis2box trigger continuous integration (CI) testing on GitHub Actions.

GitHub Actions invokes functional testing as well as integration testing to ensure regressions.

15.1.1 Unit testing

Unit tests are in `tests/unit`.

15.1.2 Integration testing

Integration tests are in `tests/integration/integration.py`.

15.1.3 Functional testing

Functional tests are defined as part of GitHub Actions in `.github/workflows/tests-docker.yml`.

15.2 Versioning

wis2box follows the Semantic Versioning Specification (SemVer).

15.3 Code Conventions

Python code follows PEP8 coding conventions.

CHAPTER

SIXTEEN

CONTRIBUTING

wis2box is developed as a free and open source project on GitHub. Contributions to the project (documentation, bug fixes, enhancements, tests, etc.) are welcome and encouraged. Please consult the wis2box Contribution guidelines for more information.

CHAPTER

SEVENTEEN

SUPPORT

Please consult the wis2box Discussions for support with the project.

CHAPTER

EIGHTEEN

LICENSE

18.1 Software

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
```

(continues on next page)

(continued from previous page)

```
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    (except as stated in this section) patent license to make, have made,
    use, offer to sell, sell, import, and otherwise transfer the Work,
    where such license applies only to those patent claims licensable
    by such Contributor that are necessarily infringed by their
    Contribution(s) alone or by combination of their Contribution(s)
    with the Work to which such Contribution(s) was submitted. If You
    institute patent litigation against any entity (including a
    cross-claim or counterclaim in a lawsuit) alleging that the Work
    or a Contribution incorporated within the Work constitutes direct
    or contributory patent infringement, then any patent licenses
    granted to You under this License for that Work shall terminate
    as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
```

(continues on next page)

(continued from previous page)

```
    Work or Derivative Works thereof in any medium, with or without
    modifications, and in Source or Object form, provided that You
    meet the following conditions:

    (a) You must give any other recipients of the Work or
        Derivative Works a copy of this License; and

    (b) You must cause any modified files to carry prominent notices
        stating that You changed the files; and

    (c) You must retain, in the Source form of any Derivative Works
        that You distribute, all copyright, patent, trademark, and
        attribution notices from the Source form of the Work,
        excluding those notices that do not pertain to any part of
        the Derivative Works; and

    (d) If the Work includes a "NOTICE" text file as part of its
        distribution, then any Derivative Works that You distribute must
        include a readable copy of the attribution notices contained
        within such NOTICE file, excluding those notices that do not
        pertain to any part of the Derivative Works, in at least one
        of the following places: within a NOTICE text file distributed
        as part of the Derivative Works; within the Source form or
        documentation, if provided along with the Derivative Works; or,
        within a display generated by the Derivative Works, if and
        wherever such third-party notices normally appear. The contents
        of the NOTICE file are for informational purposes only and
        do not modify the License. You may add Your own attribution
        notices within Derivative Works that You distribute, alongside
        or as an addendum to the NOTICE text from the Work, provided
        that such additional attribution notices cannot be construed
        as modifying the License.

    You may add Your own copyright statement to Your modifications and
    may provide additional or different license terms and conditions
    for use, reproduction, or distribution of Your modifications, or
    for any such Derivative Works as a whole, provided Your use,
    reproduction, and distribution of the Work otherwise complies with
    the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
    any Contribution intentionally submitted for inclusion in the Work
    by You to the Licensor shall be under the terms and conditions of
    this License, without any additional terms or conditions.
    Notwithstanding the above, nothing herein shall supersede or modify
    the terms of any separate license agreement you may have executed
    with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
    names, trademarks, service marks, or product names of the Licensor,
    except as required for reasonable and customary use in describing the
    origin of the Work and reproducing the content of the NOTICE file.
```

(continues on next page)

(continued from previous page)

```
7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
```

(continues on next page)

(continued from previous page)

```
    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

18.2 Documentation

The documentation is released under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

CHAPTER

NINETEEN

INDICES AND TABLES

- genindex
- modindex
- search