wis2box

Release 0.1.0

World Meteorological Organization (WMO)

2022-03-03

TABLE OF CONTENTS:

Author World Meteorological Organization (WMO)

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

Release 0.1.0

Date 2022-03-03

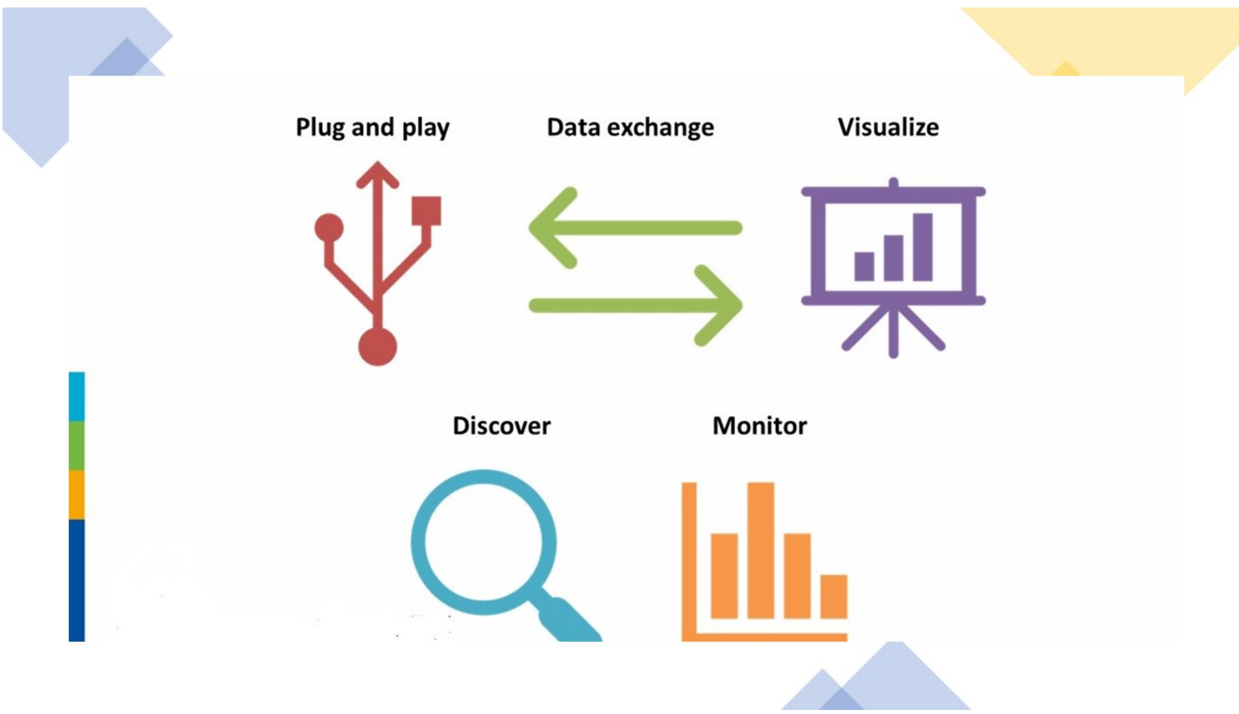
CHAPTER

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 WIS 2.0 network to your local environment
- standards-based data services and access mechanisms:
  - Data
    - BUFR
  - APIs
    - OGC API
    - MQTT
    - STAC
- 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

CHAPTER

TWO

QUICKSTART

Download wis2box and start using Malawi test data:

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

For the purposes of a quickstart, this deployment expects the test environment, which includes data and metadata. This is done by using the test environment file:

```
cp tests/test.env dev.env
vi dev.env
# ensure WIS2BOX_HOST_DATADIR is set to a local path on disk for persistent storage
```

Note: For more information on deployment, see *Administration* and *Configuration*

Start wis2box with Docker Compose and login to the wis2box container:

```
python3 wis2box-ctl.py start
python3 wis2box-ctl.py status --all # The --all flag shows all containers, even ones␣
↪that are down.
python3 wis2box-ctl.py login
```

Once logged in, create the enviroment and verify it is correct:

```
wis2box environment create
wis2box environment show
```

Setup observation data processing and API publication:

```
wis2box data setup --topic-hierarchy data.core.observations-surface-land.mw.FWCL.
↪landFixed
wis2box api add-collection --topic-hierarchy data.core.observations-surface-land.mw.FWCL.
↪landFixed $WIS2BOX_DATADIR/metadata/discovery/surface-weather-observations.yml
```

Publish station collection and discovery metadata to the API:

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

Process data via CLI:

```
wis2box data ingest --topic-hierarchy data.core.observations-surface-land.mw.FWCL.
↪landFixed --path $WIS2BOX_DATADIR/observations/WIGOS_0-454-2-AWSNAMITAMBO_2021-07-07.
↪csv
wis2box api add-collection-items --recursive --path $WIS2BOX_DATADIR/data/public
```

Logout of wis2box container:

```
exit
```

Restart the wis2box API container:

```
python3 wis2box-ctl.py restart pygeoapi
```

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

THREE

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

FOUR

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 integrate their data holdings and publish them to the WMO Information System with a plug and play capability supporting data publishing, discovery and access.

4.1 High level system context

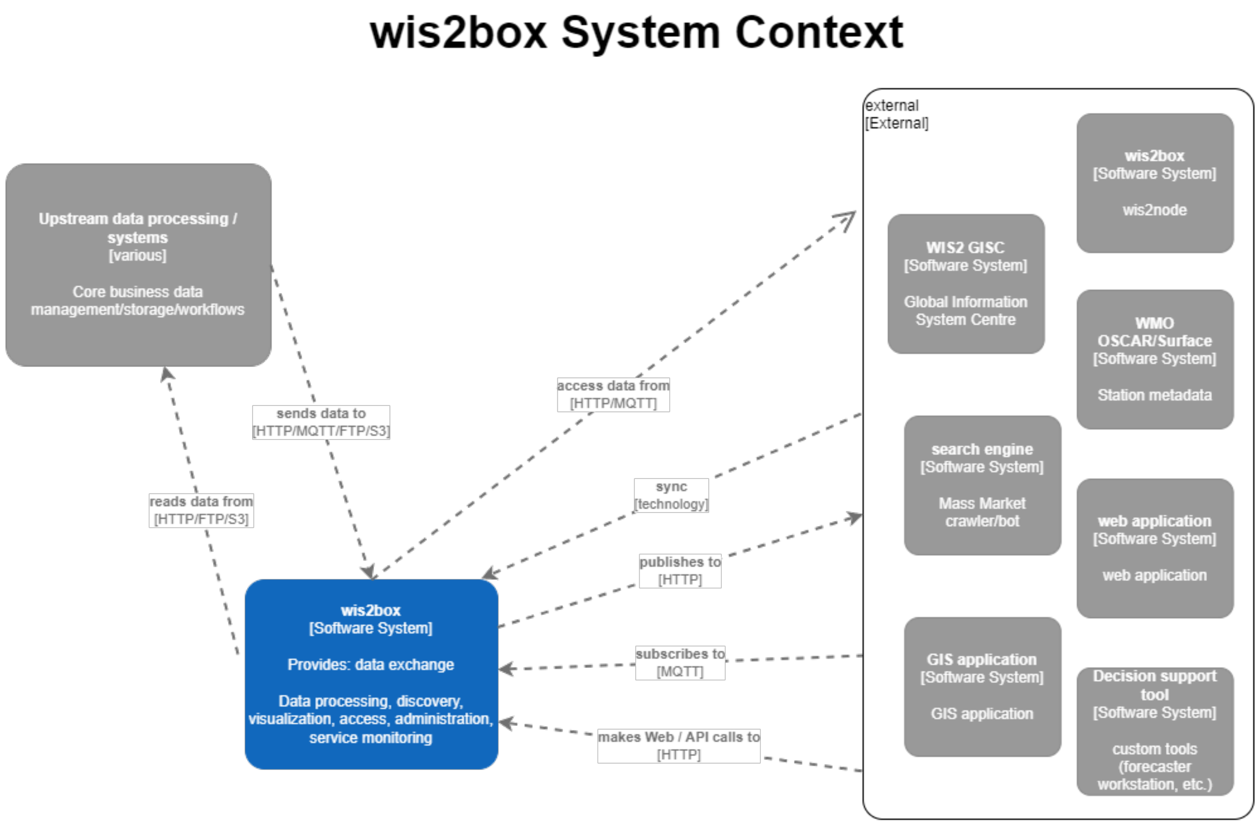
The following diagram provides a high level overview of the main functions of wis2box:

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

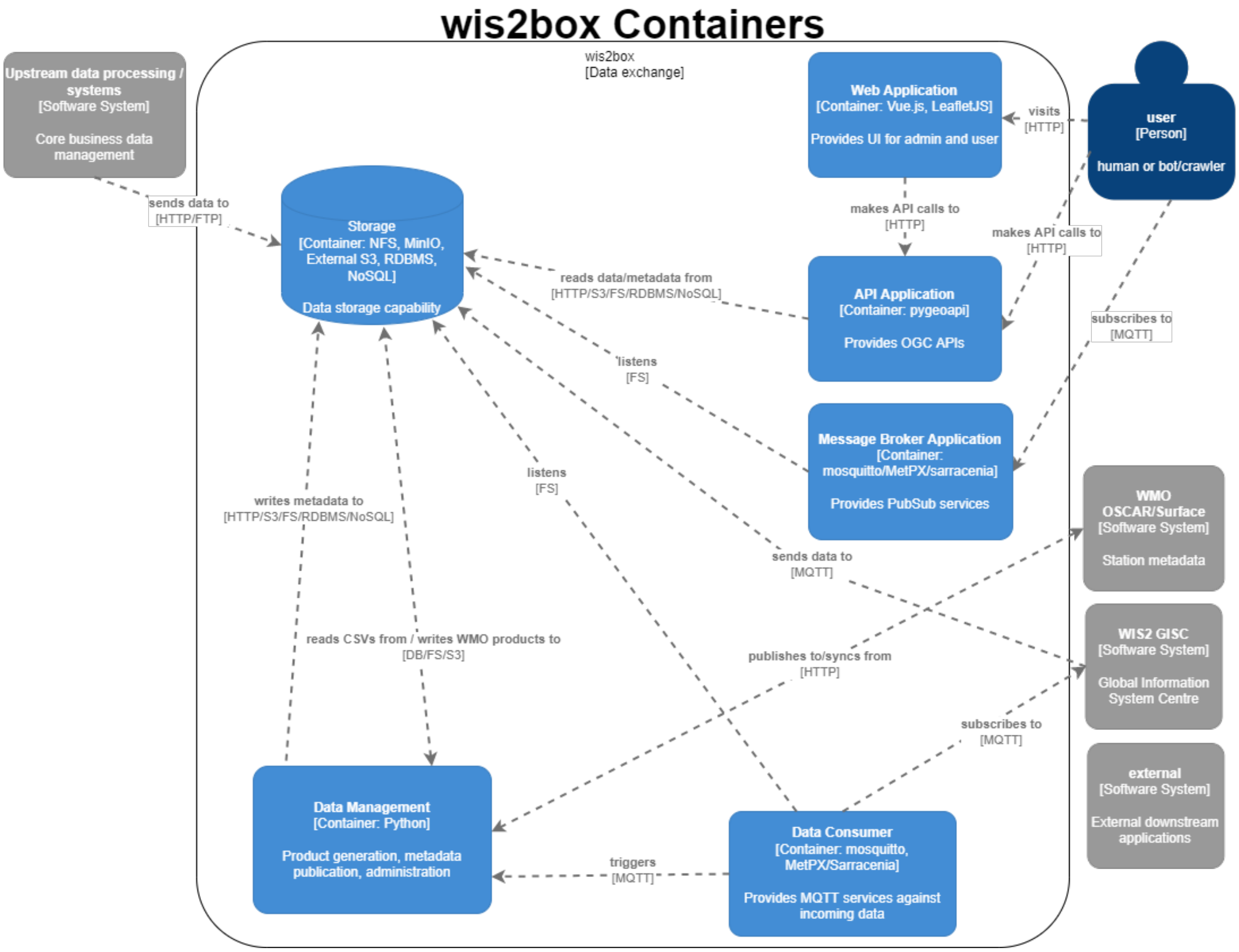
4.2 Docker Compose

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


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

4.3 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:

- **Data Consumer**: the data entry point of wis2box. Data pipelines and workflow begins here
- **Data Management**: the epicentre of wis2box. Provides core wis2box administration and data/workflow/publishing utilities
- **Storage**: core data persistence
- **API Application**: OGC APIs providing geospatial web services
- **Web Application**: user interface

4.4 Technology

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

Container	Function	Technology	Standards
Data Consumer	PubSub	mosquitto	MQTT
Data Management	data processing and publishing	pygeometa pyoscar	WCMP (WMO Core Metadata Profile) WMDR (WIGOS Metadata Record)
API Application	data discovery and access	pygeoapi Elasticsearch	OGC API
Web Application	data discovery and visualization	Vue.js Leaflet	OGC API

CHAPTER

FIVE

INSTALLATION

wis2box is built for easy installation across various operating systems and environments.

5.1 Requirements and dependencies

wis2box requires Python 3 and Docker 1.13+.

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

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

5.2 ZIP Archive

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

5.3 GitHub

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

5.4 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

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:

- **Data**: locations of where data is stored as well as retention specifications
- **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 Data

The data configurations provide control of directories on the host machine bound into the Docker volume and wis2box. The default relationship below resembles the directory structure within the wis2box volume.

Note: Make sure to use **absolute paths** instead of relative paths.

```
WIS2BOX_HOST_DATADIR=${PWD}/wis2box-data # wis2box host data directory
WIS2BOX_DATADIR=/data/wis2box  # wis2box data directory
WIS2BOX_DATA_RETENTION_DAYS=7  # wis2box data retention time, in days. Data older than␣
↪this value is
                               # is deleted on a daily basis
```

6.2.2 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_CONFIG=${PWD}/docker/pygoeoapi/pygeoapi-config.yml  # configuration file
WIS2BOX_API_BACKEND_TYPE=Elasticsearch  # backend provider type
WIS2BOX_API_BACKEND_URL=http://elasticsearch:9200  # internal backend connection URL
```

6.2.3 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
```

6.2.4 PubSub

PubSub configuration provides connectivity information for the PubSub broker.

```
WIS2BOX_BROKER=mqtt://wis2box:wis2box@mosquitto/  # RFC 1738 syntax of internal broker␣
↪endpoint
```

6.2.5 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
```

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:

```
# Required
# Host machine data directory path
WIS2BOX_HOST_DATADIR=/path/to/local/data/directory

# Optional
# Environment variable overrides
```

```
# data paths and retention
WIS2BOX_DATADIR=/data/wis2box
WIS2BOX_DATA_RETENTION_DAYS=7

# API
WIS2BOX_API_TYPE=pygeoapi
WIS2BOX_API_URL=http://localhost:8999/pygeoapi
WIS2BOX_API_CONFIG=/data/wis2box/pygeoapi-config.yml
WIS2BOX_API_BACKEND_TYPE=Elasticsearch
WIS2BOX_API_BACKEND_URL=http://elasticsearch:9200

# logging
WIS2BOX_LOGGING_LOGLEVEL=ERROR
WIS2BOX_LOGGING_LOGFILE=stdout

# PubSub
WIS2BOX_BROKER=mqtt://wis2box:wis2box@mosquitto

# 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}/wis2box-data-mappings.yml
```

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.

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 API/PubSub publishing

8.3 Running topics

8.3.1 Environment

wis2box requires the environment to be initialized before data processing or publishing.

```
wis2box environment create
```

This command will create all the directories required. You can check the environment at any time with:

```
wis2box environment show
```

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

8.3.2 Concepts

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

- **topic hierarchy**: thesaurus defined by WMO to categorize and classify data, allowing for easy and efficient search
- **discovery metadata**: description of a dataset to be included in the WIS 2.0 global 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

8.3.3 Topic hierarchy

Note: The WIS 2.0 topic hierarchies are 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, by implementing directory hierarchies. For example, the below exemplifies a WIS 2.0 topic hierarchy as implemented in wis2box:

WIS 2.0 topic hierarchy	wis2box directory
`foo.bar.baz`	`foo/bar/baz`

wis2box topic hierarchies are managed **under** the various wis2box directories, and are used as part of both design time and runtime workflow.

To create a wis2box topic hierarchy:

```
wis2box data setup --topic-hierarchy foo.bar.baz
```

This will create the topic hierarchy under the required wis2box directories in support of automated processing and publishing.

To view a given topic hierarchy setup:

```
wis2box data info --topic-hierarchy foo.bar.baz
```

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:

```
data:
    data.core.observations-surface-land.mw.FWCL.landFixed: wis2box.data.observations.
↪ObservationData
```

The format of the `data` property is `key: value`, where:

- `key`: the topic hierarchy defined in the system

- `value`: the codepath that implements the relevant data processing

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_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
Balaka,0-454-2-AWSBALAKA
Kayerekera,0-454-2-AWSKAYEREKERA
Lobi_EPA,0-454-2-AWSLOBI
Malomo_EPA,0-454-2-AWSMALOMO
Namitambo,0-454-2-AWSNAMITAMBO
Nkhoma_University,0-454-2-AWSNKHOMA
Toleza,0-454-2-AWSTOLEZA
```

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.

8.3.5.1 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.

```yaml
wis2box:
    retention: P30D
    topic_hierarchy: data.core.observations-surface-land.mw.FWCL.landFixed
    data_category: observationsSurfaceLand
    country_code: mw
    originator: FWCL
    station_type: landFixed

mcf:
    version: 1.0

metadata:
    identifier: data.core.observations-surface-land.mw.FWCL.landFixed
    language: en
    language_alternate: fr
    charset: utf8
    hierarchylevel: dataset
    datestamp: 2021-11-29

spatial:
    datatype: vector
    geomtype: point

identification:
    language: en
    charset: utf8
    title:
        en: Surface weather observations (hourly)
    abstract:
        en: Surface weather observations (hourly)
    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:
```

(continues on next page)

(continued from previous page)


```yaml
                    - 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
    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.

8.3.6.1 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 GeoJSON features.

8.3.7.1 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
```

8.3.7.2 Event driven ingest, processing and publishing

One all metadata, topic hierarchies, and data configurations are setup, event driven workflow will immediately start to listen on files in `WIS2BOX_DATADIR/data/incoming` as they are placed in the appropriate topic hierarchy directory.

Note: wis2box can make `WIS2BOX/data/incoming` accessible via webdav by enabling `docker/docker-compose.webdav.yml`.

8.3.7.3 Summary

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

8.3.8 API publishing

At this stage:

- 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.

8.3.8.1 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
```

8.3.8.2 Discovery metadata API publishing

This step will publish dataset discovery metadata to the API.

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

8.3.8.3 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 api add-collection $WIS2BOX_DATADIR/data/config/foo/bar/baz/discovery-metadata.
↪yml --topic-hierarchy foo.bar.baz
```

To delete the colection from the API backend and configuration:

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

Note that the data itself is being published to the API backend automatically given the event driven workflow. If manual ingest is needed, the following command can be run in interactive mode:

```
wis2box api add-collection-items --topic-hierarchy foo.bar.baz
```

8.3.8.4 API container restart

Any change to API configuration requires a restart of the API container, which can be run via the following:

```
python3 wis2box-ctl.py restart wis2box
```

8.3.8.5 Summary

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

8.3.9 Data retention

wis2box is configured to set data retention according to your requirements. Data retention is managed via the `WIS2BOX_DATA_RETENTION_DAYS` environment variable as part of configuring wis2box. Data retention includes cleaning of published data and archiving of incoming/raw data.

8.3.9.1 Cleaning

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

```
# delete data older than WIS2BOX_DATA_RETENTION_DAYS by default
wis2box data clean


# delete data older than --days (force override)
wis2box data clean --days=$WIS2BOX_DATA_RETENTION_DAYS
```

8.3.9.2 Archiving

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

```
wis2box data archive
```

Data is archived to `WIS2BOX_DATADIR/data/archive`.

CHAPTER

NINE

SERVICES

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

9.1 OGC API

wis2box data and metadata are made available via the OGC API - Features and OGC API - Records standards.

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

TODO: example requests

9.2 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.

9.3 Web Accessible Folder (WAF)

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

http://localhost:8999/data/

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

9.4 MQTT

The wis2box MQTT endpoint can be found at:

mqtt://localhost:1883

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

CHAPTER

TEN

DATA ACCESS

10.1 Overview

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

10.2 API

10.2.1 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.

[106]:
```
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/pygeoapi'
```

10.2.1.1 Stations

Let's find all the stations in our wis2box:

[107]:
```
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: 19
Stations:

BALAKA
BILIRA
CHIDOOLE
CHIKANGAWA
CHIKWEO
CHINGALE
KASIYA AWS
KASUNGU NATIONAL PARK AWS
KAWALAZI
KAYEREKERA
LENGWE NATIONAL PARK
LOBI AWS
MAKANJIRA
MALOMO
MLOMBA
MTOSA BENGA
NAMITAMBO
NKHOMA UNIVERSITY
TOLEZA
```

10.2.1.2 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.

[108]:
```
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.observations-surface-land.mw.FWCL.landFixed, title: Surface weather␣
↪observations (hourly)
```

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

[109]:
```
dataset_id = 'data.core.observations-surface-land.mw.FWCL.landFixed'

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

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

(continues on next page)

(continued from previous page)

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

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

```
Data access links:

http://localhost:8999/pygeoapi/collections/data.core.observations-surface-land.mw.FWCL.
↪landFixed (OAFeat)
mqtt://mosquitto/ (MQTT)
```

[109]:
```
['http://localhost:8999/pygeoapi/collections/data.core.observations-surface-land.mw.FWCL.
↪landFixed',
 'mqtt://mosquitto/']
```

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

[110]:
```
dataset_api_link = [link['href'] for link in response['associations'] if link['type'] ==
↪'OAFeat'][0]

dataset_api_link
```

[110]:
```
'http://localhost:8999/pygeoapi/collections/data.core.observations-surface-land.mw.FWCL.
↪landFixed'
```

10.2.1.3 Observations

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

[111]:
```
url = f'{dataset_api_link}/items'

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

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

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


```
{
  "id": "WIGOS_0-454-2-AWSCHIDOOLE_20220119T125500",
  "conformsTo": [
    "http://www.opengis.net/spec/ogcapi-features-1/1.0/req/geojson",
    "http://www.wmo.int/spec/om-profile-1/-/req/geojson"
  ],
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [
      34.5,
```


(continues on next page)

(continued from previous page)


```
      -15.47,
      929.0
    ]
  },
  "properties": {
    "identifier": "WIGOS_0-454-2-AWSCHIDOOLE_20220119T125500",
    "phenomenonTime": "2022-01-19T12:55:00+00:00",
    "resultTime": "2022-02-21T15:27:56+00:00",
    "wigos_station_identifier": "0-454-2-AWSCHIDOOLE",
    "metadata": [
      {
        "name": "height_of_station_above_ground_level",
        "value": 929.0,
        "units": "m"
      }
    ],
    "observations": {
      "air_pressure": {
        "value": 90903.14,
        "units": "Pa",
        "metadata": [
          {
            "name": "sensor_height_above_mean_sea_level",
            "value": 930.0,
            "units": "m"
          }
        ]
      },
      "pressure_at_mean_sea_level": {
        "value": 101623.7,
        "units": "Pa",
        "metadata": [
          {
            "name": "sensor_height_above_mean_sea_level",
            "value": 930.0,
            "units": "m"
          }
        ]
      },
      "change_of_air_pressure_over_past_3_hours": {
        "value": null,
        "units": "Pa",
        "metadata": [
          {
            "name": "sensor_height_above_mean_sea_level",
            "value": 930.0,
            "units": "m"
          }
        ]
      },
      "characteristic_of_pressure_tendency": {
        "value": 4.0,
```


(continues on next page)

(continued from previous page)


```
        "units": "CODE TABLE",
        "metadata": [
          {
            "name": "sensor_height_above_mean_sea_level",
            "value": 930.0,
            "units": "m"
          }
        ]
      },
      "air_temperature": {
        "value": 24.25,
        "units": "Celsius",
        "metadata": [
          {
            "name": "sensor_height_above_local_ground",
            "value": 1.5,
            "units": "m"
          }
        ]
      },
      "dew_point_temperature": {
        "value": 21.25,
        "units": "Celsius",
        "metadata": [
          {
            "name": "sensor_height_above_local_ground",
            "value": 1.5,
            "units": "m"
          }
        ]
      },
      "relative_humidity": {
        "value": 83.0,
        "units": "%",
        "metadata": [
          {
            "name": "sensor_height_above_local_ground",
            "value": 1.5,
            "units": "m"
          }
        ]
      },
      "duration_of_sunshine_1hr": {
        "value": 0.0,
        "units": "min",
        "metadata": [
          {
            "name": "time_period",
            "value": -1.0,
            "units": "h"
          }
        ]
```


(continues on next page)

(continued from previous page)


```
    },
    "duration_of_sunshine_24h": {
      "value": 0.0,
      "units": "min",
      "metadata": [
        {
          "name": "time_period",
          "value": -24.0,
          "units": "h"
        }
      ]
    },
    "precipitation_amount_1h": {
      "value": 0.0,
      "units": "kg m-2",
      "metadata": [
        {
          "name": "time_period",
          "value": -1.0,
          "units": "h"
        },
        {
          "name": "sensor_height_above_local_ground",
          "value": 1.5,
          "units": "m"
        }
      ]
    },
    "air_temperature_maximum": {
      "value": 24.550000000000001,
      "units": "Celsius",
      "metadata": [
        {
          "name": "cell_methods",
          "description": "maximum"
        },
        {
          "name": "time_period_start",
          "value": -24.0,
          "units": "h"
        },
        {
          "name": "time_period_end",
          "value": 0.0,
          "units": "h"
        },
        {
          "name": "sensor_height_above_local_ground",
          "value": 1.5,
          "units": "m"
        }
      ]
```


(continues on next page)

(continued from previous page)


```
    },
    "air_temperature_minimum": {
      "value": 23.650000000000034,
      "units": "Celsius",
      "metadata": [
        {
          "name": "cell_methods",
          "description": "minimum"
        },
        {
          "name": "time_period_start",
          "value": -24.0,
          "units": "h"
        },
        {
          "name": "time_period_end",
          "value": 0.0,
          "units": "h"
        },
        {
          "name": "sensor_height_above_local_ground",
          "value": 1.5,
          "units": "m"
        }
      ]
    },
    "wind_from_direction": {
      "value": 104.0,
      "units": "deg",
      "metadata": [
        {
          "name": "cell_methods",
          "value": 2.0,
          "units": "CODE TABLE"
        },
        {
          "name": "time_period",
          "value": -10.0,
          "units": "min"
        },
        {
          "name": "sensor_height_above_local_ground",
          "value": 2.0,
          "units": "m"
        },
        {
          "name": "wind_sensor_type",
          "value": 0.0,
          "units": "FLAG TABLE"
        }
      ]
    },
```


(continues on next page)

(continued from previous page)


```
    "wind_speed": {
      "value": 0.878,
      "units": "m/s",
      "metadata": [
        {
          "name": "cell_methods",
          "value": 2.0,
          "units": "CODE TABLE"
        },
        {
          "name": "time_period",
          "value": -10.0,
          "units": "min"
        },
        {
          "name": "sensor_height_above_local_ground",
          "value": 2.0,
          "units": "m"
        },
        {
          "name": "wind_sensor_type",
          "value": 0.0,
          "units": "FLAG TABLE"
        }
      ]
    },
    "wind_speed_maximum_gust": {
      "value": 2.64,
      "units": "m/s",
      "metadata": [
        {
          "name": "cell_methods",
          "value": null,
          "units": "CODE TABLE"
        },
        {
          "name": "time_period",
          "value": null,
          "units": "min"
        },
        {
          "name": "sensor_height_above_local_ground",
          "value": 2.0,
          "units": "m"
        },
        {
          "name": "wind_sensor_type",
          "value": 0.0,
          "units": "FLAG TABLE"
        }
      ]
    },
```


(continues on next page)

(continued from previous page)


```
      "surface_downwelling_shortwave_flux_in_air_1h": {
        "value": 287336.3,
        "units": "J m-2",
        "metadata": [
          {
            "name": "cell_methods",
            "description": "sum"
          },
          {
            "name": "time_period",
            "value": -1.0,
            "units": "h"
          }
        ]
      },
      "surface_downwelling_shortwave_flux_in_air_24h": {
        "value": 287336.3,
        "units": "J m-2",
        "metadata": [
          {
            "name": "cell_methods",
            "description": "sum"
          },
          {
            "name": "time_period",
            "value": -24.0,
            "units": "h"
          }
        ]
      }
    },
    "id": "WIGOS_0-454-2-AWSCHIDOOLE_20220119T125500"
  }
}
```


Let's inspect what's measured at Chidoole:

[112]:
```python
print('Observed properties:\n')
for key, value in response['features'][0]['properties']['observations'].items():
    print(f'{key} ({value["units"]})')
```

```
Observed properties:

air_pressure (Pa)
pressure_at_mean_sea_level (Pa)
change_of_air_pressure_over_past_3_hours (Pa)
characteristic_of_pressure_tendency (CODE TABLE)
air_temperature (Celsius)
dew_point_temperature (Celsius)
relative_humidity (%)
duration_of_sunshine_1hr (min)
duration_of_sunshine_24h (min)
precipitation_amount_1h (kg m-2)
```

(continues on next page)

(continued from previous page)

```
air_temperature_maximum (Celsius)
air_temperature_minimum (Celsius)
wind_from_direction (deg)
wind_speed (m/s)
wind_speed_maximum_gust (m/s)
surface_downwelling_shortwave_flux_in_air_1h (J m-2)
surface_downwelling_shortwave_flux_in_air_24h (J m-2)
```

10.2.2 Pandas

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

[113]:
```
import pandas as pd

datestamp = [obs['properties']['phenomenonTime'] for obs in response['features']]
air_temperature = [obs['properties']['observations']['air_temperature']['value'] for obs␣
↪in response['features']]

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

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

[114]:
```
df
```

[114]:
```
                     Date/Time  Air temperature (°C)
0    2022-01-19T12:55:00+00:00                 24.25
1    2022-01-19T13:55:00+00:00                 25.35
2    2022-01-19T14:55:00+00:00                 24.55
3    2022-01-19T15:55:00+00:00                 23.45
4    2022-01-19T16:55:00+00:00                 21.95
..                         ...                   ...
151  2022-01-29T10:55:00+00:00                 27.05
152  2022-01-29T11:55:00+00:00                 29.95
153  2022-01-29T12:55:00+00:00                 28.55
154  2022-01-29T13:55:00+00:00                 27.35
155  2022-01-29T14:55:00+00:00                 22.35

[156 rows x 2 columns]
```

[115]:
```
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

```

(continues on next page)

(continued from previous page)

```
Begin: 2022-01-19T12:55:00+00:00
End: 2022-01-29T14:55:00+00:00
Summary statistics:
```

[115]:

```
       Air temperature (°C)
count            156.000000
mean              22.708974
std                2.764659
min               16.650000
25%               20.725000
50%               22.250000
75%               25.075000
max               29.950000
```

10.2.3 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:

[13]:

```
from owslib.ogcapi.features import Features

import pandas as pd

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


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

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

[14]:

```
oafeat = Features(api)

collections = oafeat.collections()
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('data.core.observations-surface-land.mw.FWCL.
↪landFixed')
malawi_obs_df = pd.DataFrame(malawi_obs['features'])

# then filter by station
namitambo_obs = oafeat.collection_items('data.core.observations-surface-land.mw.FWCL.
↪landFixed', wigos_station_identifier='0-454-2-AWSNAMITAMBO')
namitambo_obs_df = pd.DataFrame(namitambo_obs['features'])
print(malawi_obs_df.dtypes)
print(malawi_obs_df.head(3))
```

```
This OGC API Features endpoint has 3 datasets
Surface weather observations (hourly)
Stations
Discovery metadata
id            object
conformsTo    object
type          object
geometry      object
properties    object
dtype: object
                                        id  \
0  WIGOS_0-454-2-AWSBALAKA_20220114T075500
1  WIGOS_0-454-2-AWSBALAKA_20220114T085500
2  WIGOS_0-454-2-AWSBALAKA_20220114T095500

                                          conformsTo     type  \
0  [http://www.opengis.net/spec/ogcapi-features-1...  Feature
1  [http://www.opengis.net/spec/ogcapi-features-1...  Feature
2  [http://www.opengis.net/spec/ogcapi-features-1...  Feature

                                            geometry  \
0  {'type': 'Point', 'coordinates': [34.97, -14.9...
1  {'type': 'Point', 'coordinates': [34.97, -14.9...
2  {'type': 'Point', 'coordinates': [34.97, -14.9...

                                          properties
0  {'identifier': 'WIGOS_0-454-2-AWSBALAKA_202201...
1  {'identifier': 'WIGOS_0-454-2-AWSBALAKA_202201...
2  {'identifier': 'WIGOS_0-454-2-AWSBALAKA_202201...
```

10.2.4 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://pygeoapi/pygeoapi" # jupyter is run through docker
#oapi = http://localhost:8999/pygeoapi # 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'
```

(continues on next page)

(continued from previous page)

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

    filter, lag


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

    intersect, setdiff, setequal, union
```

10.2.4.1 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_id             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...
```

10.2.4.2 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>,
```

(continues on next page)

(continued from previous page)

```
#   X_metadata.anytext <chr>, id <chr>, geometry <POLYGON [°]>
```

10.2.4.3 Observations

[4]:
```
malawi_obs <- read_sf(paste0(oapi,"/collections/data.core.observations-surface-land.mw.
↪FWCL.landFixed/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
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 [°]>
```

[ ]:
```
```

10.3 PubSub

10.3.1 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.

[13]:
```
import json
import paho.mqtt.client as mqtt
import random
import urllib
import urllib.request


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

(continues on next page)

(continued from previous page)

```
r = random.Random()
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.

[14]:
```
def sub_connect(client, userdata, flags, rc, properties=None):
    print("on connection to subscribe: ", mqtt.connack_string(rc))
    for s in ["xpublic/#"]:
        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: `xpublic/#`, 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.

[15]:
```
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:

[16]:
```
def getData(m, sizeMaximum=1000):
    """
    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:

[18]:
```
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.
message 0 topic: xpublic/v03/WIS/us/mobile_rgnl_al/surface/aviation/metar/us received: {
↪'mode': '664', 'mtime': '20220224T052208.259097815', 'atime': '20220224T052208.
↪259097815', 'pubTime': '20220224T052208.264983', 'baseUrl': 'http://localhost:8999/
↪data/20220224T05', 'relPath': 'WIS/us/mobile_rgnl_al/surface/aviation/metar/us/SAUS44_
↪KMOB_240503_COR_8d674aab16213ac2b13fab2d79950456.txt', 'integrity': {'method': 'md5',
↪'value': 'jWdKqxYhOsKxP6steZUEVg=='}, 'size': 137}
message 0 data: b'SAUS44 KMOB 240503 COR\r\r\nMTRPRN\r\r\nMETAR KPRN 240458Z AUTO␣
↪20006G15KT 10SM OVC006 19/16 A3016 RMK AO2 \r\r\nSLP161 T01940161 402830183\r\r\n\r\r\
↪n\x03'
message 1 topic: xpublic/v03/WIS/pr/tjgu/surface/miscellaneous/pr received: {'mode': '664
↪', 'mtime': '20220224T052208.427098989', 'atime': '20220224T052208.427098989', 'pubTime
↪': '20220224T052208.430775', 'baseUrl': 'http://localhost:8999/data/20220224T05',
↪'relPath': 'WIS/pr/tjgu/surface/miscellaneous/pr/SXPU52_TJGU_240418_
↪a8f650c50a0c0e38a41b0867a011574f.txt', 'integrity': {'method': 'md5', 'value':
↪'qPZQxQoMDjikGwhnoBFXTw=='}, 'size': 67}
message 1 data: b'SXPU52 TJGU 240418\r\r\nAAXX 24044\n78523 35/// /0503   30151 222//\r\
↪r\n\x03'
message 2 topic: xpublic/v03/WIS/ca/canadian_met_centre/upperair/aircraft/airep/north-
↪atlantic received: {'mode': '664', 'mtime': '20220224T052209.0511043072', 'atime':
↪'20220224T052209.0511043072', 'pubTime': '20220224T052209.056451', 'baseUrl': 'http://
↪localhost:8999/data/20220224T05', 'relPath': 'WIS/ca/canadian_met_centre/upperair/
↪aircraft/airep/north-atlantic/UANT01_CWAO_240503_2d512e655e32ce80001105dfa2fc19f0.txt',
↪ 'integrity': {'method': 'md5', 'value': 'LVEuZV4yzoAAEQXfovwZ8A=='}, 'size': 135}
message 2 data: b'UANT01 CWAO 240503\r\r\nARP BAW17V 5329N04306W 0503 F400 5400N04000W␣
↪0515 5400N03000W MS70\r\r\n 260/88 KT\r\r\nGZBKN DDL XXH 240503 L48A\r\r\n\r\r\n\x03'
```


(continues on next page)

(continued from previous page)


```
message 3 topic: xpublic/v03/WIS/pr/tjgu/surface/miscellaneous/pr received: {'atime':
↪'20220224T052208.435099125', 'mtime': '20220224T052208.435099125', 'mode': '664',
↪'pubTime': '20220224T052208.440895', 'baseUrl': 'http://localhost:8999/data/20220224T05
↪', 'relPath': 'WIS/pr/tjgu/surface/miscellaneous/pr/SXPU52_TJGU_240413_
↪63e3ff1d1e3bc11b1f430024622ae5aa.txt', 'integrity': {'method': 'md5', 'value': 'Y+P/
↪HR47wRsfQwAkYirlqg=='}, 'size': 67}
message 3 data: b'SXPU52 TJGU 240413\r\r\nAAXX 24044\n78523 35/// /0404   30151 222//\r\
↪r\n\x03'
message 4 topic: xpublic/v03/WIS/us/wallops_i__wallops_station_va/surface/miscellaneous/
↪nc received: {'mode': '664', 'atime': '20220224T052208.44309926', 'mtime':
↪'20220224T052208.44309926', 'pubTime': '20220224T052208.445723', 'baseUrl': 'http://
↪localhost:8999/data/20220224T05', 'relPath': 'WIS/us/wallops_i__wallops_station_va/
↪surface/miscellaneous/nc/SXNC50_KWAL_240503_99baec43c8b040b9e8496a762be9a891.txt',
↪'integrity': {'method': 'md5', 'value': 'mbrsQ8iwQLnoSWp2K+mokQ=='}, 'size': 132}
message 4 data: b'SXNC50 KWAL 240503\r\r\n\x1e326A9318 055050324 \r\n07.54 \r\n002 \r\
↪n120 \r\n038 \r\n041 \r\n100 \r\n13.0 \r\n027.0 \r\n347 \r\n005 \r\n00000 \r\n 44+0NN ␣
↪28W\r\r\n\x03'
message 5 topic: xpublic/v03/WIS/pr/tjgu/surface/miscellaneous/pr received: {'mode': '664
↪', 'mtime': '20220224T052208.455099344', 'atime': '20220224T052208.455099344', 'pubTime
↪': '20220224T052208.457988', 'baseUrl': 'http://localhost:8999/data/20220224T05',
↪'relPath': 'WIS/pr/tjgu/surface/miscellaneous/pr/SXPU52_TJGU_240403_
↪0034251607312a5feff05fd760128747.txt', 'integrity': {'method': 'md5', 'value':
↪'ADQlFgcxKl/v8F/XYBKHRw=='}, 'size': 67}
message 5 data: b'SXPU52 TJGU 240403\r\r\nAAXX 24044\n78523 35/// /0306   30151 222//\r\
↪r\n\x03'
```


```
[18]: 7
```

10.4 Running These 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.

10.5 Summary

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

CHAPTER

ELEVEN

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.

11.1 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:

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

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

    def transform(self, input_data: Path) -> bool:
        # transform data
        # populate self.output_data with a dict as per:
        self.output_data [{
            '_meta': {
                'identifier': 'c123'
                '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 should provide a `list` of objects with the following properties:

- `_meta`: object with identifier and Python datetime objects based on the observed datetime of the data

- `<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

11.2 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.

11.3 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-malawi-observations

11.4 Example plugins

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

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

CHAPTER

TWELVE

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.

12.1 Testing

12.1.1 Unit testing

TODO

12.1.2 Integration testing

TODO

12.1.3 Functional testing

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

12.2 Versioning

wis2box follows the Semantic Versioning Specification (SemVer).

12.3 Code Conventions

Python code follows PEP8 coding conventions.

CHAPTER

THIRTEEN

CONTRIBUTING

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

CHAPTER FOURTEEN

SUPPORT

Please consult the wis2box Discussions for support with the project.

CHAPTER

FIFTEEN

LICENSE

15.1 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)

15.2 Documentation

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

CHAPTER SIXTEEN

INDICES AND TABLES

- genindex
- modindex
- search