WOUDC Data Registry

Tom Kralidis

Jun 28, 2022

CONTENTS:

1 Overview **1**

2 Installation **3**
2.1 Requirements . . . 3
2.2 Instructions . . . 3

3 Configuration **5**

4 Administration **7**
4.1 Creating Backends . . . 7
4.2 Deleting Backends . . . 7
4.3 Populating Backends . . . 8

5 Running **9**
5.1 Model Interface . . . 9
5.2 Ingestion . . . 12
5.3 Verification . . . 12
5.4 UV Index generation . . . 12
5.5 Total Ozone table generation . . . 13
5.6 OzoneSonde table generation . . . 13
5.7 Peer data centre indexing . . . 13

6 Operator Workflows **15**
6.1 Gathering . . . 15
6.2 Processing . . . 15
6.3 Notifications . . . 16
6.4 Publishing . . . 16

7 Publishing **17**

8 Development **19**
8.1 Dataset Table Definitions . . . 19

9 Support **23**

10 License **25**

11 Reference **27**

12 Indices and tables **29**

CHAPTER

ONE

OVERVIEW

CHAPTER

TWO

INSTALLATION

The WOUDC Data Registry is available on GitHub.

2.1 Requirements

- Python 3 and above
- virtualenv
- ElasticSearch 5.5.0 and above

Other requirements are as listed in requirements.txt, and are installed during project installation.

2.2 Instructions

- **Create and activate a virtual environment::**

  ```
  virtualenv woudc-data-registry
  cd woudc-data-registry
  source bin/activate
  ```

- **Install the project::**

  ```
  git clone https://github.com/woudc/woudc-data-registry.git
  cd woudc-data-registry

  pip install -r requirements.txt # Core dependencies
  pip install -r requirements-pg.txt # For PostgreSQL backends

  python setup.py build
  python setup.py install
  ```

- **Set up the project::**

  ```
  . /path/to/environment/config.env # Set environment variables

  # Continue by creating databases as per instructions in Administration
  ```

CHAPTER

THREE

CONFIGURATION

The WOUDC Data Registry is configured mostly using environment variables. Operators will be provided with environment files which set these variables appropriately. Executing one of these environment files is required before running the WOUDC Data Registry.

Some of the environment variables are paths to more specific configuration files. All these paths are interpreted on the machine where the Data Registry is running, and most of the files are included with the WOUDC Data Registry project in the **/data** and **/data/migrate** folders.

Any configuration options can be changed directly in the environment after running an environment file.

WDR_LOGGING_LOGLEVEL
Minimum severity of log messages written to the log file.

WDR_LOGGING_LOGFILE
Full path to log file location.

WDR_DB_DEBUG
Whether to include log messages from the database in logging output.

WDR_DB_TYPE
DBMS used to host the Data Registry (e.g. sqlite, postgresql).

WDR_DB_HOST
HTTP URL to Data Registry DB host, or filepath to Sqlite DB file if applicable.

WDR_DB_PORT
Port number for Data Registry DB host, if applicable.

WDR_DB_NAME
Name of Data Registry schema within the DB.

WDR_DB_USERNAME
Username for Data Registry DB user account.

WDR_DB_PASSWORD
Password for Data Registry DB user account.

WDR_SEARCH_TYPE
DBMS used to host the Search Index (e.g. elasticsearch)

WDR_SEARCH_URL
HTTP URL to Search Index host.

WDR_WAF_BASEDIR
Path to WAF files system location on the host machine.

WDR_WAF_BASEURL
HTTP URL to WAF location on the web.

WDR_TABLE_SCHEMA
Path to JSON schema for table definition file, on the host machine.

WDR_TABLE_CONFIG
Path to table definition file on the host machine.

The file defines the structure of tables expected in Extended CSV input to the Data Registry, including those that must appear in every Extended CSV file as well as those that appear only with specific datasets.

All tables have a list of required and optional fields, a range of allowable numbers of rows, and a range of allowable number of appearances in the file. File metadata must be sufficient to identify which table definitions must or may show up in the file.

WDR_ERROR_CONFIG
Path to error definition file on the host machine.

The file defines error types and messages and their severity. All entries listed as type Error in this file cause the WOUDC Data Registry to stop processing an input. Entries of type Warning may be recovered from, and the Data Registry may be able to process a file regardless of any Warnings it receives.

Warnings and Errors are logged in the Operator Report as part of the WOUDC Data Registry's core workflow.

WDR_ALIAS_CONFIG
Path to alias configuration file on the host machine.

The file defines alternate spellings for certain fields in input files. If encountered, any of these alternate spellings are substituted for one standard spelling, unless this substitution is marked as type Error in the error definitions file.

To display all configurations from woudc-data-registry, run *woudc-data-registry admin config*

CHAPTER

FOUR

ADMINISTRATION

In addition to the project code itself, WOUDC Data Registry depends on several databases which must be set up before the project can run.

The core backend, referred to as the Data Registry, is an archive of file metadata and links using a relational database, currently PostgreSQL. The Data Registry is the trusted and long-term source of file metadata, history and versioning, and file location.

The secondary backend, known as the Search Index, is derived from the Data Registry and is currently kept in ElasticSearch. The Search Index is denormalized and will contain additional metrics data for quick access.

The whole file contents, as opposed to metadata only, are stored in the WOUDC Web Accessible Folder (WAF) and are linked via URLs in both the Data Registry and Search Index.

The Search Index is intended for use by web applications and for searchability, while the Data Registry is intended for internal operations and to prop and compare against the Search Index in case of error.

4.1 Creating Backends

make ENV=/path/to/environment/config.env createdb
Create the Data Registry database instance

woudc-data-registry admin registry setup
Create Data Registry tables and schema

woudc-data-registry admin registry setup
Create Search Index mappings

4.2 Deleting Backends

woudc-data-registry admin registry teardown
Delete all Data Registry tables and schema

woudc-data-registry admin search setup
Delete all Search Index indexes and mappings

make ENV=/path/to/environment/config.env dropdb
Delete the Data Registry database instance

4.3 Populating Backends

When necessary, the entire history of data records can be recreated by processing all files in the WAF (backfilling). Not all metadata can be extracted from these files, and so there are alternate methods to recreate metadata from scratch.

If the Data Registry is empty, its metadata is recovered from a series of files. The WOUDC Data Registry code comes with two files in the data/init directory:

- ships.csv
- countries.json

The .csv and .json files contain core metadata for the Data Registry. Contact WOUDC to provide the remaining required metadata files.

After ensuring all metadata files are together in one directory, run:

woudc-data-registry admin init -d <initialization> <flags>
: Searches the directory path <initialization> for .csv and .json files of metadata, and loads them into the Data Registry tables. If the –init-search-index flag is provided, loads them to the Search Index as well.

If the Data Registry is filled but not the Search Engine, the latter can be populated by using the sync command (see *Publishing*).

CHAPTER

FIVE

RUNNING

5.1 Model Interface

The model interface allows for direct access to the Data Registry's contents from the command line for certain types of metadata records.

Operations include viewing records, adding or deleting records, and updating existing records.

Main use cases for the interface are to add missing metadata records or new records during a processing run, or to fix small numbers of out-of-date or damaged records between processing runs.

woudc_data_registry project list
List all overarching projects in the Data Registry.

woudc_data_registry dataset list
List all datasets received by the Data Registry.

woudc_data_registry station list
List all stations that participate in the WOUDC project.

woudc_data_registry contributor list
List all contributors to the WOUDC project.

woudc_data_registry instrument list
List all instruments that have recorded data for WOUDC submissions.

woudc_data_registry deployment list
List all contributor deployments that have submitted to the WOUDC project.

woudc_data_registry station add <options>
Add a station record to the Data Registry, with metadata as specified by options:

-id IDENTIFIER WOUDC station identifier.
-gi GAW_ID GAW station identifier.
-n NAME Name of the station.
-t TYPE Type of the station or ship.
-g GEOMETRY Latitude, longitude, and elevation of the station.
-c COUNTRY Country where the station is located.
-w WMO_REGION_ID WMO identifier for station's continent/region.
-sd START_DATE Date when the station became active.
-ed END_DATE Date when the station became inactive.

woudc_data_registry contributor add <options>

Add a WOUDC contributing agency to the Data Registry, with metadata as specified by options:

-n (name) Full name of the contributor
-a (acronym) Abbreviated name of the contributor
-p (project) Overarching project that the contributor is a member of
-g (geometry) Latitude and longitude for the contributor's headquarters
-c (country) Contributor's country of origin or country of operation
-w (wmo_region) WMO identifier for contributor's home continent/region
-u (url) HTTP URL to the contributor's home webpage
-e (email) Address for central contact to the contributing agency
-f (ftp_username) Username for the contributor in the WOUDC FTP

woudc_data_registry instrument add <options>

Add an instrument to the Data Registry with metadata as specified by options:

-n (name) Name of the producing company or source of the instrument
-m (model) Instrument model name
-s (serial) Instrument serial number
-d (dataset) Type of data the instrument is able to record
-st (station) Station from where the instrument is operated
-g (geometry) Latitude and longitude where the instrument is located

woudc_data_registry deployment add <options>

Add a contributor deployment to the Data Registry with metadata as specified by options:

-c (contributor) ID of the contributor in the Data Registry
-s (station) Station WOUDC ID where the contributor is operating
-sd (start_date) Date when the deployment started
-ed (end_date) Date when the deployment ended

woudc_data_registry station update -id <identifier> <options>

Modify an existing station record in the Data Registry with the ID <identifier> as specified by options:

-gi (gaw_id) GAW station identifier
-n (name) Name of the station
-t (type) Type of the station or ship
-g (geometry) Latitude, longitude, and elevation of the station
-c (country) Country where the station is located
-w (wmo_region) WMO identifier for station's continent/region
-sd (start_date) Date when the station became active
-ed (end_date) Date when the station became inactive

woudc_data_registry contributor update -id <identifier> <options>

Modify an existing contributor record in the Data Registry with the ID <identifier>, and possibly change it to a new ID, as specified by options:

-n (name) Full name of the contributor
-a (acronym) Abbreviated name of the contributor
-p (project) Overarching project that the contributor is a member of
-g (geometry) Latitude and longitude for the contributor's headquarters
-c (country) Contributor's country of origin or country of operation
-w (wmo_region) WMO identifier for contributor's home continent/region
-u (url) HTTP URL to the contributor's home webpage
-e (email) Address for central contact to the contributing agency
-f (ftp_username) Username for the contributor in the WOUDC FTP

woudc_data_registry instrument update -id <identifier> <options>

Modify an existing instrument record in the Data Registry with the ID <identifier>, and possibly change it to a new ID, as specified by options:

-n (name) Name of the producing company or source of the instrument
-m (model) Instrument model name
-s (serial) Instrument serial number
-d (dataset) Type of data the instrument is able to record
-st (station) Station from where the instrument is operated
-g (geometry) Latitude and longitude where the instrument is located

woudc_data_registry deployment update -id <identifier> <options>

Modify an existing contributor deployment record in the Data Registry with the ID <identifier>, and possibly change it to a new ID, as specified by options:

-c (contributor) ID of the contributor in the Data Registry
-s (station) Station WOUDC ID where the contributor is operating
-sd (start_date) Date when the deployment started
-ed (end_date) Date when the deployment ended

woudc_data_registry station|contributor|instrument|deployment show <id>

Display all information in the Data Registry about the record which has the identifier <id> under the specified metadata type (station, contributor, instrument, or deployment)

woudc_data_registry station|contributor|instrument|deployment delete <id>

Delete the record with identifier <id> from the Data Registry, under the specified metadata type (station, contributor, instrument, or deployment).

5.2 Ingestion

The primary workflow involving the WOUDC Data Registry is ingestion, or bulk processing, of input files. Ingest commands sequentially parse, validate, repair, break down and upload contents of these files to the Data Registry as well as the Search Index. A copy of the incoming file is sent to the WAF.

woudc_data_registry data ingest <input_source> <flags>

Ingest the incoming data at <input_source>, which is either a path to a single input file or to a directory structure containing them. Output log files and reports are placed in <working_dir>. <flags> are as follows:

-y (yes) Automatically accept all permission checks

-l (lax) Only validate core metadata tables, and not dataset-specific metadata or data tables. Useful when data is presented in old formats or is formatted improperly and cannot be repaired but must be ingested anyways, such as during backfilling.

5.3 Verification

A secondary workflow in the WOUDC project is input file verification or error-checking. As a mock ingestion, the same logging output is released as in ingestion (including to the console) but no changes are made to the Data Registry or Search Engine backends.

This workflow finds whether files are properly formatted, which can inform contributors whether their file generation processes and their metadata are correct. WOUDC Data Registry developers may also use the verification command to test ingestion routines on dummy input files without inserting dummy data into the backends.

woudc_data_registry data verify <input_source> <flags>

Verify the incoming data at <input_source>, which is either a path to a single input file or to a directory structure containing them. <flags> are as follows:

-y (yes) Automatically accept all permission checks

-l (lax) Only validate core metadata tables, and not dataset-specific metadata or data tables. Useful when only core tables and metadata are important or when dataset-specific tables are known to contain errors but nothing can be done about them, such as during backfilling.

5.4 UV Index generation

An hourly UV Index can be generated using data and metadata from WOUDC Extended CSV data. In particular, files from the Broadband and Spectral datasets are used in this process. The UV Index can be generated from a single process to build the entire index.

woudc-data-registry product uv-index generate <flags> /path/to/archive/root

Delete all records from the uv_index_hourly table and use all Spectral and Broadband files to generate uv_index_hourly records. <flags> are as follows:

-y (yes) Automatically accept all permission checks

woudc-data-registry product uv-index update <options> <flags> /path/to/archive/root

Only generate uv_index_hourly table entries using files within a year range. No records are deleted. If a start or end year is not specified, then no lower or upper bound will be used within the year range. <flags> are as follows.

-y (yes) Automatically accept all permission checks

<options are as follows.

-sy (start-year) Optional lower bound of year range

-ey (end-year) Optional upper bound of year range

5.5 Total Ozone table generation

A Total Ozone table can be generated using data and metadata from WOUDC TotalOzone Extended CSV files. This table provides a more detailed representation of TotalOzone data and allows specific measurements to be directly queried.

woudc-data-registry product totalozone generate <flags> /path/to/archive/root

Delete all records from the totalozone table and use all TotalOzone files to generate totalozone records. <flags> are as follows:

-y (yes) Automatically accept all permission checks

5.6 OzoneSonde table generation

An OzoneSonde table can be generated using data and metadata from WOUDC OzoneSonde Extended CSV files. This table provides a more detailed representation of OzoneSonde data and allows specific measurements to be directly queried.

woudc-data-registry product totalozone generate <flags> /path/to/archive/root

Delete all records from the totalozone table and use all TotalOzone files to generate totalozone records. <flags> are as follows:

-y (yes) Automatically accept all permission checks

5.7 Peer data centre indexing

WOUDC supports indexing the following remote data centres:

- Eubrewnet
- NDACC

woudc-data-registry peer eubrewenet index -fi /path/to/data-centre-file-index.txt

woudc-data-registry peer ndacc index -fi /path/to/data-centre-file-index.txt

CHAPTER

SIX

OPERATOR WORKFLOWS

Every Monday the WOUDC Data Registry is updated with new files submitted over the past week from various contributors. Files are either successfully ingested immediately, have recoverable errors that are fixed manually, or have errors that are irrecoverable and fail to process. Contributors are alerted to their failing files and given feedback on their errors, and may attempt to resubmit the files later.

6.1 Gathering

TODO

6.2 Processing

In the processing stage, incoming files go through a series of validation checks and are assessed for formatting and metadata errors. Files that pass are chunked and stored in the Data Registry, the DEV Search Engine, and the WAF and become publicly visible.

The operator's role is to help assess and respond to errors that the WOUDC Data Registry program can't deal with, such as correcting formatting or correcting bad metadata, until as many of the incoming files can be persisted as possible.

Setup:

- Start a new screen session using *screen -S processing-<currentdate>*
- Switch to (or create) the WOUDC Data Registry master virtual environment
- Create a working directory at */apps/data/incoming-<currentdate>-run*
- Run the master environment file to set environment variables and configurations

Processing:

- Attempt to process all incoming files from the week:
- *woudc_data_registry data ingest /apps/data/incoming/<currentdate> -w /apps/data/incoming-<currentdate>-run*
- Watch for the occasional prompt as a new instrument, station name, or contributor deployment is found.
- Some files will fail to be processed because of a recoverable or irrecoverable error. Recoverable errors can be fixed between runs and reprocessed, but files with irrecoverable errors need not be processed again until the contributor resubmits them later.
- To reprocess a selection of failed files:
- Copy the selection of files to a new directory, named *failing_files*, within the working directory and run:

- *woudc_data_registry data ingest /apps/data/incoming-<currentdate>-run/failing_files*
- And then continue to reprocess failures from these runs using the second command.
- Once a file has been processed successfully, do not move it to a new location!

6.3 Notifications

- Create a file named */apps/data/incoming-<currentdate>-run/failed-files-<currentdate>*. This file can be created and edited while processing is going on.
- The file contains one block for each contributor that submitted data in the previous week, and begins with the contributor's acronym in all uppercase, a space, and a semicolon-separated email list between parentheses. Following the header is a summary block as shown here.
- Fill in the file with pass/fix/fail counts for each contributor. Also, in each contributor block, document error messages and the files they affected in Summary of Fixes and Summary of Failures blocks.
- Contributor email can be fetched using a SQL query to the Data Registry like this one:
- *SELECT acronym, email FROM contributors JOIN deployments USING (contributor_id) WHERE station_id = '…'*
- Where … is replaced with the #PLATFORM.ID from any of the input files from that contributor.
- #PLATFORM.ID is used in the query because most files get it right, while some files have the wrong agency name. If the #PLATFORM.ID seems wrong, feel free to query the contributor using a different field. than other fields.
- Another operator will use this email report as the basis for a feedback email message to each contributor.

6.4 Publishing

TODO

CHAPTER

SEVEN

PUBLISHING

Currently, processing a file updates the Search Index with its metadata as long as the file passes validation. This way the Search Index and Data Registry should stay synchronized as long as no manual changes are made.

In case the Data Registry and Search Index become desynchronized, there is a command to resync them. This command is destructive and cannot be undone, so use with caution and only when the Data Registry's content is trusted.

woudc-data-registry admin search sync

Synchronize the Data Registry and Search Index exactly, using the Data Registry as a template. First inserts or update records in the Index to match the row with the same ID in the Registry, and then deletes excess documents in the Index with no ID match in the Registry.

In the future, whether to add elements to the Search Index while processing may turn into a configuration option.

CHAPTER

EIGHT

DEVELOPMENT

8.1 Dataset Table Definitions

Tables in Extended CSV files must conform to a certain format, and certain tables are required for each dataset. The specifications for table formatting are defined in table definition files, two of which are provided with the WOUDC Data Registry codebase.

data/tables.yml is a table definition file for production use when processing incoming files. **data/migrate/tables-backfilling.yml** is an alternative table definition file for backfilling the registry with historical files. The second file is looser on required tables and fields to allow flexibility with older WOUDC formats.

8.1.1 Table Definition Format

Table definition files are made up of organized sections of table definitions. Each table definition is a dictionary-type element (in a JSON or YAML file or equivalent) which defines the shape and expected fields of a table.

A table definition is formatted like this (example in YAML):

```
table_name:
    rows: <range of rows>
    occurrences: <range of occurrences>
    ~required_fields:
        - list
        - of
        - required
        - fields
    ~optional_fields:
        - list
        - of
        - optional
        - fields
```

Optional keys are prefixed with the ~ character.

In a table definition, a range of integers **a** < **b** is specified by a string with one of the following forms:

```
b      The discrete number b.
a-b    The range of integers between a and b (inclusive).
b+     The range of integers with no upper bound starting at b.
```

The **rows** key defines the allowable height range for the table, i.e. the number of rows the table must have.

The **occurrences** key defines the allowable range of number of times the table appears in the file. If 0 is included in the range then the table as a whole is optional and may be left out of a file.

The (optional) **required_fields** key defines a list of field names that must appear in the table. If any of these fields is missing from the file, the table will be considered invalid. All tables must have at least one required field.

The (optional) **optional_fields** key defines a list of field names that may appear in the table but are not required.

8.1.2 Dataset Definition Format

The set of table definitions varies based on which type of data the file contains. Table definition files express this by organizing table definitions into sections.

Some tables are common to all Extended CSV files. The **Common** key at the top of a table definition files contains the definitions for all these shared tables.

Besides the common tables, the table definitions that apply to an Extended CSV file depend on certain fields within the file.

The table definitions apply depend on the file's dataset (the type of data contained, controlled by the **#CONTENT.Category** value), its level (of QA, controlled by the **#CONTENT.Level** value), and its form (controlled by its **#CONTENT.Form** value). In some cases there are multiple allowable table definitions for the same dataset, level, and form, in which case each option is assigned an integer version number starting from 1.

A dataset definition is formatted like this (example in YAML):

```
dataset_name:
  "level":
    "form":
      table definition 1
      table definition 2
      ...
      data_table: <name of data table>
    ..
  ..
```

OR:

```
dataset_name:
  "level":
    "form":
      "version":
        table definition 1
        table definition 2
        ...
        data_table: <name of data table>
      ..
    ..
  ..
```

The numeric level, form, and version keys all must be surrounded by double quotes to force them to be strings. There may be multiple level keys in a dataset, multiple forms within a level, and multiple versions within a form.

In the innermost block is a series of table names mapped to table definitions. The one additional key, data_table, maps to the name of the table containing observational data. This must correspond to a required table amongst the table definitions in that block.

8.1.3 Asserting Correctness

The WOUDC Data Registry process validates all table definition files against a schema before using them. The schema is written using JSON schema language and is stored at **data/table-schema.json**.

WOUDC developers adding to a table definitions file can use JSON schema validation tools to check that their additions are in the right format.

CHAPTER

NINE

SUPPORT

CHAPTER

TEN

LICENSE

CHAPTER

ELEVEN

REFERENCE

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search