––––

Upper Air Database v0

Issued by:

Date:

Ref:

Official reference number service contract:

This document has been produced in the context of the Copernicus Climate Change Service (C3S).

The activities leading to these results have been contracted by the European Centre for Medium-Range Weather Forecasts, operator of C3S on behalf of the European Union (Delegation Agreement signed on 11/11/2014). All information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose.

The user thereof uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission

and the European Centre for Medium-Range Weather Forecasts has no liability in respect of this document, which is merely representing the authors view.

Contributors

UNIVIE

F. Ambrogi

M. Blaschek

L. Haimberger

# Introduction

This document provides details regarding the Upper Air Database v0 .

With the term database we define a set of netCDF files, one for each observation station, that include radiosondes observation data from several original sources.

While several sources offer data for the same station, datasets are often incomplete,

i.e. they might lack of some observation time intervals, or some specific data might be missing. The aim of this work is indeed to collect the largest amount of information from several available dataset, and build a comprehensive database of all the available information. This include the observation data as well as any auxiliary metadata that can help interpret the meaning of the data, for example details about the sensors of the sondes or of the location of the observing stations.

This database will be soon delivered to the Copernicus Data Store, so that the necessary adjustments and interfaces with the CDS webpage can be implemented in order to make the dataset available for download and use.

The files are structured according the Common Data Model (CDM) that can be consulted on the GitHub page:

<https://github.com/glamod/common_data_model/>

In the rest of the document, we will describe the source of the data, how the data have been merged together in one unique file, the proposed structure of the output file, a brief description of the extensions and improvements we envisage for version v1, and a quick guide on how to access the data from the netCDF files.

# Data Sources

The data were contained in seven different original sources, that for simplicity are labelled as ERA5\_1, ERA5\_1759, ERA5\_1761, ERA5\_3188, BUFR, IGRA2, NCAR.

***ERA5\_1***:

***ERA5\_1759***:

***ERA5\_1761***:

***ERA5\_3188***:

***BUFR***:

***NCAR***:

***IGRA2***:

… description of the datasets …

The structure of the file of each dataset is similar. In particular, we call “record” the set of data that was acquired during each launch of the radiosonde. Every record contains measurements of several variables at different pressure levels. In the CDM language, this information is stored in the ‘report\_id’ variable e.g. in the ‘header\_table’ or in the ‘observations\_table’. The NCAR files, this number is reported in the header of each record.

For the other datasets, this information is missing. It can be constructed by assigning a progressive number to each different record contained in each file, looping through it in chronological order. In a similar fashion, also a counter identifying each single observation of each single variable, at each pressure level, at each date and time can be defined. This constitute the variable ‘observation\_id’ in the CDM tables. Note that, as it will be discussed in Section xxx, these two numbers are necessary to identify the source of each data in the merged files.

# Merging Procedure

Each of the dataset described in the previous section might contain data for a certain station. The aim of what we call merging procedure is to combine information taken from different sources, and different input files, into a single merged output file, in netCDF format and compliant with the CDM requirements. The aim of this Section is to extensively describe the merging procedure we use in order to combine the observation data from the different datasets.

To clarify the concept of merging, as a trivial example, we consider the case where only one dataset is available for a certain observation:

(i.e. selection of a given measured variable, for a given pressure level) for a specific observation date and time. Note that with the term “data” we refer to the value of a meteorological variables (*temperature, wind speed, wind direction, relative humidity, geopotential*), reported at a specific date and time, for a specific value of the pressure (called *pressure level*).

For this trivial case, the merging procedure simply reduces to the selection of the data from the only possible original dataset. Now let's consider the case of two datasets which have data for the same variable, i.e. they report a measured value for a certain observable, for the same pressure level, acquired on the same date and time. Ideally both datasets will report the same value; however, it might happen than the values were, for example, rounded and the data from a certain dataset reports more significant digits than the other. Another possibility is that data were removed from a dataset since the corresponding measurements were flagged as not accurate, while the data might still appear in other datasets. Moreover, data might have been interpolated to standard pressure levels, so that in one dataset there might be more pressure levels available.

It is crucial to define in an unambiguous way how the data to enter the final merged file are selected, and allow to reconstruct their original source. Note that the merging procedure only concerns data belonging to the same report, i.e. for each available report only one dataset will be chosen, or in other words, there will be no combination of different dataset for different pressure levels, or different measured values for different variables in a given report: the data belonging to a defined report must come only from a single dataset.

We know describe how the merging procedure is performed.

Merging of the observation date and time

The first parameter to merge is the date and time ("date\_time" in the observations\_tables). Every record from each dataset has its own time stamp with the date and time. This might be slightly different from one dataset to the other; for the v0 of the database, we neglect any time difference and classify as different observations also those one which fall within a modest time difference, e.g. 1 or 2 hours, which in principle are the same observation (since typically one have only one launch per station every day, or at most every 6 hours). This is a check left for future versions of the database.

We then extract the list of “date\_time” for each dataset, and from those we create a combined list, that constitute the global set of available temporal observations.

Merging of the pressure levels observations

For each distinct *date\_time*, we have data from several dataset. We then loop over the available pressure levels, which might be different for each dataset. For each pressure level, a set of measurements for the different meteorological variables is available.

1. Code Overview and Workflow

Here we describe briefly the structure of the merging\_cdm\_netCDF.py script and the main workflow for merging the different files;extended details as well as usage examples can be found in the code, either as in-line comments or doc-strings.

The script uses the following external modules:

- netCDF4 for reading netCDF files;

- numpy;

- pandas;

- h5py;

- xarray,

in addition to other standard python modules such as datetime, warnings and logging.

By default the logging level is set to INFO, but it can be easily changed to print additional debugging statements by replacing the line

*logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')*

with

*logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')*

at the top of the code.

The core of the code consists of a class called Merging, that retains relevant variables stored as attributes, and contains the methods used to process the files.

The class is initialized in the main section of the scrip. Then, two methods are called:

- *initialize\_data()*, which essentially reads the input netCDF files (in CDM compliant form, produced by the data harvester code described in deliverable XXX)

and adapts the input data to be easily processed

- *merge()*, which analyses the input data and produce a merged file.

The only necessary input is a dictionary where each key is the name of the dataset, and the key is the complete path to the source netCDF file:

*example\_input = { 'ncar\_w' : 'example\_stations/ncar/chuadb\_windc\_82930.txt.nc',*

*'ncar\_t' : 'example\_stations/ncar/chuadb\_windc\_82930.txt.nc',*

*'igra2' : 'example\_stations/igra2/chBRM00082930-data.txt.nc',*

*'era5\_1' : 'example\_stations/era5\_1/chera5.conv.\_82930.nc',*

*... }*

Note that there might be two distinct ncar files, since from the original dataset, files containing temperature-related variables (i.e. upper air temperature,

dew point, humidity) and wind-related variables (wind speed, direction) are kept separated, and likewise treated from the harvester script.

Nevertheless the merging procedure will combine the data from the two distinct files and label it as 'ncar'.

**Data initialization**

The *initialize\_data* method reads the relevant information from the source netCDF files such as the the source\_configuration (which contain the name of the original source file), the station\_configuration and observations\_table tables, plus the recordtimestamp and recordindex variables.

The recordtimestamp variable is the set of unique date-time values for each observation, which characterise each sonde ascent and/or data acquisition; the recordindex variable stores the index of the first occurence of such observations inside the netCDF file. This is very practical to extract data relative to specific observation times. In fact, from the desired time interval, it is possible to extract the lower and upper recordindex mapping to the interval, and then it is sufficient to read the data (as e.g. xarray or python lists) between [recordindex\_lower:recordindex\_upper] to select the desired data, avoiding loading or reading the entire dataset.

Note that in the following we will sometimes refer to each recordtimestamp with the term date\_time.

After the loading process, the first step for the merging procedure regards the extractions of all the distinct date\_time from the different dataset.

This is done by the make\_all\_datetime method, which extracts the unique list of date\_times values from all the datasets; the recordindex of each distinct date\_time in the original dataset is stored. This way, it is known if and at which index in the original file each date\_time is available.

**Merging Criteria**

On the one hand, merging different datasets means to combine together different data, for example creating an extended time series of observations of non-overlapping data-taking periods. On the other hand, the same record, i.e. data relative to a specific date\_time, might appear in several datasets, so a series of clear rules that enable to select one preferred data source must be defined.

For this aim, the merging procedure is essentially based on the loop over all the possible merged date\_time entries, created in the initial step.

For each distinct time step, the available data must then be checked.

If only one dataset is available, there is no real operation to be performed, rather the original data must be copied into the output file.

If more than one dataset is available, then:

1. The dataset with the largest number of data (i.e. measurements at different pressure levels) is selected;

2. The dataset with the order of preference igra2, ncar, era5\_1 is selected;

3. any of the era5\_3188, era5\_1759, era5\_1761 dataset is selected.

The selection is performed by the *merge\_record* method. This can be easily modified or extended if a more accurate selection is required.

**Reanalyses Feedback**

The era\_1 dataset is the only dataset for which reanalysis information is available. The extension of the CDM tables to retain such useful data, following the proposal described in XXX, has been recently approved by XXX . For the current version v0 of the database, we make use of a simpler way to store the information. As described in the XXX report, the harvester tool stores the content of the original source files in a group called 'era5fb'. We remind that in the case of the era5 datasets (era5\_1, era5\_3188, era5\_1759, era5\_1761), the era5fb group matches exactly the whole content of the original odb file, where each column takes the name from one of the variables in the odb file, e.g. 'date@hdr','time@hdr', etc.

For each record in the merged file, it is possible to obtain the reanalysis information whenever available: if the era5\_1 record is selected, then the corresponding block from the era5fb is copied into the merged file. Otherwise, the information is empty. For completeness, we describe how this is technically implemented. To keep the merged era5fb group uniform in structure, we do the following. If, for a given station, the era5\_1 data is available, the code reads the era5fb from the source file, and creates a dataframe with a column for each variable. If the era5\_1 dataset is chosen as the best merging candidate, then the table will be filled with the proper era5fb feedback value. Otherwise, if other datasets are choosen, the same dataframe structure is kept, but filled with empty values.

This causes a problem if the era5\_1 dataset is totally missing for the station, since the era5fb cannot be read and the dataframe cannot be created. However, since there are no reanalyses data, we create an empty, single column dataframe, so that the era5fb group will be present in the merged output file, but contains no useful information.

The era5fb group is structured so that is exactly mapped to the observations\_tables entries, i.e. it shares the same recordtimestamps and recordindex.

**Observation\_id Renumbering**

The “observation\_is” variable must be unique for each observation value in the “observations\_table”. To be consistent with this requirement, but at the same time to allow to retain the original observation\_id value from the original dataset, we implemented a convertion from the original value to the merged one in the following way. We define a dataset-numbering mapping as:

IGRA2: 1 , NCAR: 2 , BUFR: 3, ERA5\_1: 4 , ERA5\_1759: 5, ERA5\_1761: 6, ERA5\_3188 :7

We then multiply this number times 1 billion, and we add it to the original ‘observation\_id’ . This way, the first digit on the merged id allows to identify the original dataset, while the digits following the zeros allow to obtain the original id.

For example, the observation\_id 789 from the era5\_1 dataset of a certain station, would become 4000000789 in the merged file. The first digit 4 identifies the era5\_1 origin, and the digits following the series of zeros match the original observation\_id .

# Output File

In this Section we describe the structure of the files that form the v0 of the database. The files will be extended and will include more metadata in the future versions. In the present form, they contain all the relevant information regarding the observation data, as well as the necessary auxiliary information to identify the original source of each entry.

We take as an example the "*merged\_chuadb\_windc\_82930.txt.nc*" file.

Each station file contains three CDM tables, i.e. the “*station\_configuration*”, the “*source\_configuration*” and “*observation\_table*”.

These tables are stores as netCDF groups. For the former two tables, there are distinct groups for each of the different dataset that contain observation data for that particular station, and that were merged together in the *observation\_table*.

As an example, the example file contains can be accessed like:

import os,sys

import netCDF4 as ncdate\_time

F ='merged\_chuadb\_windc\_82930.txt.nc'

ds = nc.Dataset(F)

and printing the variable ds will show the included groups:

bufr\_source\_configuration, bufr\_station\_configuration,

era5\_1759\_source\_configuration, era5\_1759\_station\_configuration,

era5\_1\_source\_configuration, era5\_1\_station\_configuration,

igra2\_source\_configuration, igra2\_station\_configuration,

ncar\_source\_configuration, ncar\_station\_configuration,

observations\_table

This means that data for this observing station is found in the original datasets: bufr, era5\_1, era5\_1759, igra2 and ncar, and the corresponding *station\_configuration* and *source\_configuration* tables can be found.

These tables contain unique information for the station, as collected from the original source files. They do not provide metadata varying with

each observation, so they are not repeated for each different observation.

The *source\_configuration*, as of now, contains only the useful variable *'source\_file'*, which contain the name of the data source file.

Most important is certainly the observations\_table, containing the following variables:

“date\_time”, “latitude”, “longitude”, “observation\_id”, “observation\_value”, “observed\_variable”, “source\_id”, “z\_coordinate”

For example,

ds.groups[‘observations\_table’][:1]

will access the first entry of the “observations\_table”, that looks like:

1966-08-14 12:00:00 -9.37 -54.900002 5000022158 100.0 106 era5\_1759 85000.0

This must be interpreted as the observation, at date and time *1966-08-14 12:00:00*, of the wind direction (number 106) equal to 100.0 [m/s], at the station at latitude and longitude *-9.37* and *-54.900002 .* The record\_id 5000022158 is unique for the merged file, and tells that the data in the original dataset (era5\_1759: 5) are identified by the 22158 record id.