



ARCTIC PASSION

Deliverable 2.6
MeteoIO docker image and OGC WPS service

Version 0.6, 2023-12-08: TBA



Project

Arctic PASSION

EU Horizon 2020 grant agreement

101003472

Work package 2

Bringing the Arctic Data System into action

Lead beneficiary

1 - SLF/WSL

Lead author

Mathias Bavay (SLF/WSL)

Contributors

Patrick Leibersperger (SLF/WSL)

Matteo Terruzzi (SLF/WSL - DkR Srl)

Status

-

Dissemination level

PU

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101003472



Table of Contents

- 1. Introduction 3
- 2. System overview 4
- 3. Web service overview 5
 - 3.1. Anonymous data users 6
 - 3.2. Guest users 6
 - 3.3. Data owners 7
- 4. Technical implementation 8
 - 4.1. Security considerations 8
 - 4.2. Docker containers 9
 - 4.3. MeteoIO 9
 - 4.4. Inishell 11
- 5. Upcoming developments 11
- References 12

1. Introduction

Although the polar regions have a large influence on the global climate as well as provide ecosystem services for the global community, the observational data tend to be sparse for these regions [Jung et al., 2016]. Therefore, knowing which data already exist and allowing them to be reused by the global community is a cost-effective way of filling this gap ([Jung et al., 2016], [Brun et al., 2013]). This is one of the objectives of the Arctic PASSION project whose purpose is the creation and implementation of a coherent, integrated Arctic observing system^[1]

A large amount of globally available polar meteorological data comes from the scientific community ([de Rosnay et al., 2015], [Brun et al., 2013]) where the data producers are usually relatively small groups. They are responsible for the entire data acquisition process, from setting up and maintaining the stations to publishing the data. Of special relevance to work package two of the Arctic PASSION project, this leads to a data ecosystem with a very high diversity, on various levels:

- the sensors performing the data acquisition: the sensors themselves, their setup, their calibration and maintenance.
- the data format: how is each data point encoded, time representation, how is a sequence of data points encoded and stored.
- the metadata describing the data: how are the measured parameters named, how are geographical locations represented, how is the dataset linked to other resources (such as the data producers, the description of the sensors setup).

This diversity comes at a high cost: a survey conducted on various mailing-lists related to research on the cryosphere conducted for [Bavay et al., 2020] showed that almost all data is provided in formats that are not directly supported by the tools of the data users and reading the data takes at least one day of work for three-quarters of them (Figure 1).

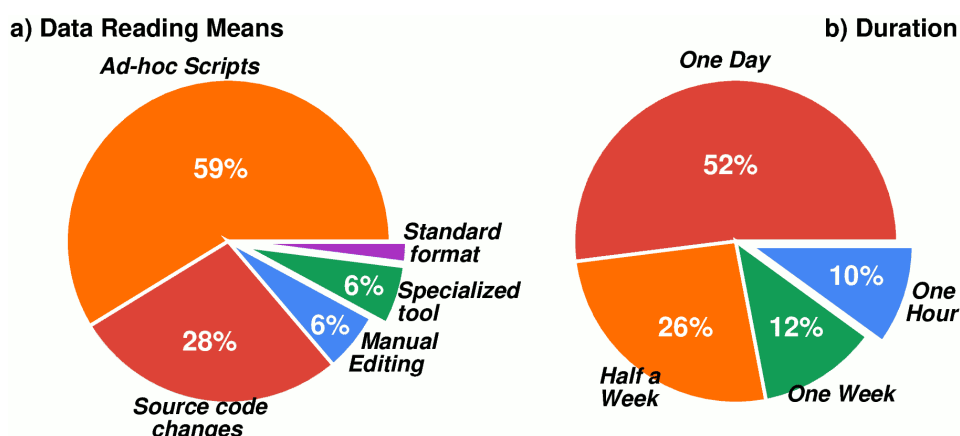


Figure 1. Survey on challenges of third party usage of cryospheric data. Panel a) shows how the respondents read third-party data while panel b) shows how long it takes to read the said third-party data - source [Bavay et al., 2020]

In order to seamlessly exchange data between systems, data level discovery and interoperability is necessary and can be achieved by relying on a set of good practices described as FAIR data management. Unfortunately, the data producers are usually research institutes which often do not have the infrastructure nor the resources to properly enable such FAIR data management^[2].

This Arctic Passion task is about designing, implementing and providing the global community with tools for the data producers to easily make their data FAIR (and not forgetting Interoperability and Reusability). This is done by standardizing the data to a common, self documenting data format (NetCDF files) conforming to well defined conventions (Climate and Forecast convention as well as with NetCDF Attribute Convention for Dataset Discovery (ACDD) metadata [Parsons et al., 2011]). This builds upon work previously done for the Global Cryosphere Watch (GCW) activity area of the World Meteorological Organization (WMO) as described in [Bavay et al., 2020].

2. System overview

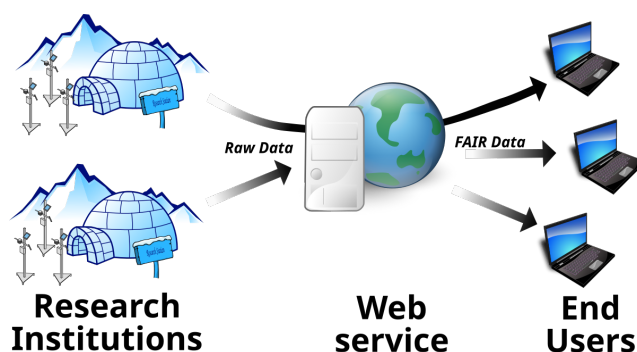


Figure 2. General concept of the web service

As shown in Figure 2, the general principle is that the data in its raw format is sent to a web service to be automatically converted into a standard-compliant, FAIR dataset. Of course, the conversion needs more information in order to proceed:

- how to understand the file format;
- how to identify the various parameters that might be present in the file;
- all the metadata that is missing in the original file also has to be provided.

This information is contained in a configuration file that follows the INI format [Bavay & Egger, 2014]. This configuration file is human readable, the goal being that in several decades our successors could easily retrieve all the information contained in such files in order to reprocess the raw files if necessary. A graphical user interface is provided in order to help creating and editing this configuration file: the Inishell tool [Bavay et al., 2022]. The processing of the data files following the instructions contained in the configuration file is performed by the MeteoIO meteorological data pre-processing library [Bavay & Egger, 2014]. The resulting data files are then either manually downloaded or can also be permanently served by the web service and published in a data catalog (Figure 3).

Because the MeteoIO library is used as data processor, it is also possible to perform data quality control (data QC) while standardizing the data. For each parameter, MeteoIO allows the user to stack an arbitrary number of data filtering and correction algorithms chosen from a toolbox of more than 40 available algorithms^[3]. MeteoIO can also log every modification applied to the data by printing for each modified data point a line to the console containing the timestamp, the station ID, the parameter name and the name of the component that altered this data point. This can then be used to as insight into possible sensor malfunctions or failure modes.

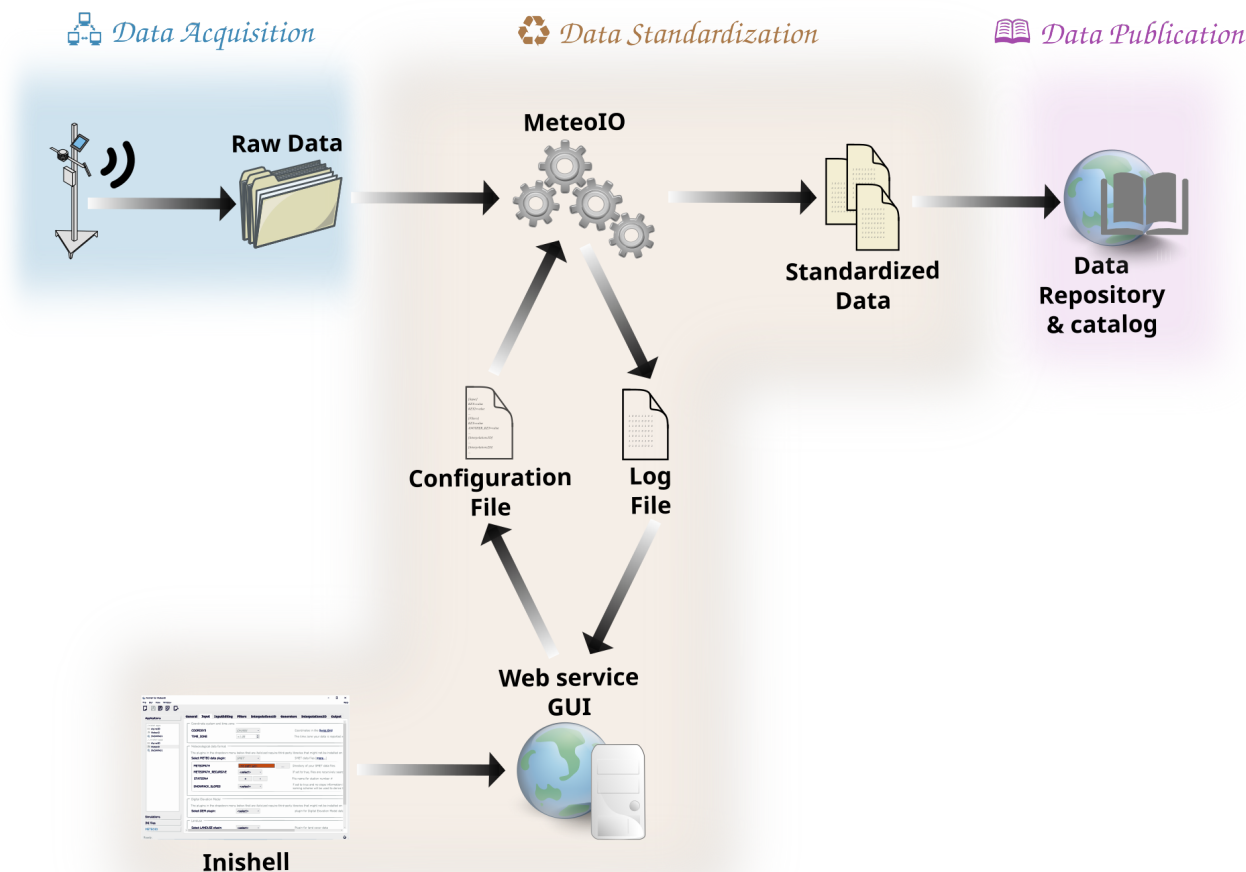


Figure 3. Components involved in web service. Please note that the data acquisition and data publication stages are out of scope for task 2.3. Nonetheless, the produced datasets can be made available through THREDDS and OpenDAP in order to be easily integrated within user-friendly data publication systems such as data portals.

Please note that in terms of licenses, every component in this system is open source: the MeteoIO library is LGPLv3^[4], the Inishell configuration interface is LGPLv3^[5], the web service is AGPLv3^[6], the THREDDS server (third party) is BSD-3-Clause^[7].

3. Web service overview

When coming to the landing page^[8] of the MeteoIO web service at <https://service-meteoio.slf.ch>, it is important to understand that the web service has three user categories:

- **anonymous data users** who browse and download standardized data provided by data owners directly in the web service interface instead of relying on third party data portals publishing this data;
- **data owners** providing their data and configuration file to be permanently stored there (usually with automatic updates to their data files) and let the system make their standardized data available;
- **guest users** who bring their data and configuration file and get a standardized file back while their input is automatically deleted after a certain time (guest job).

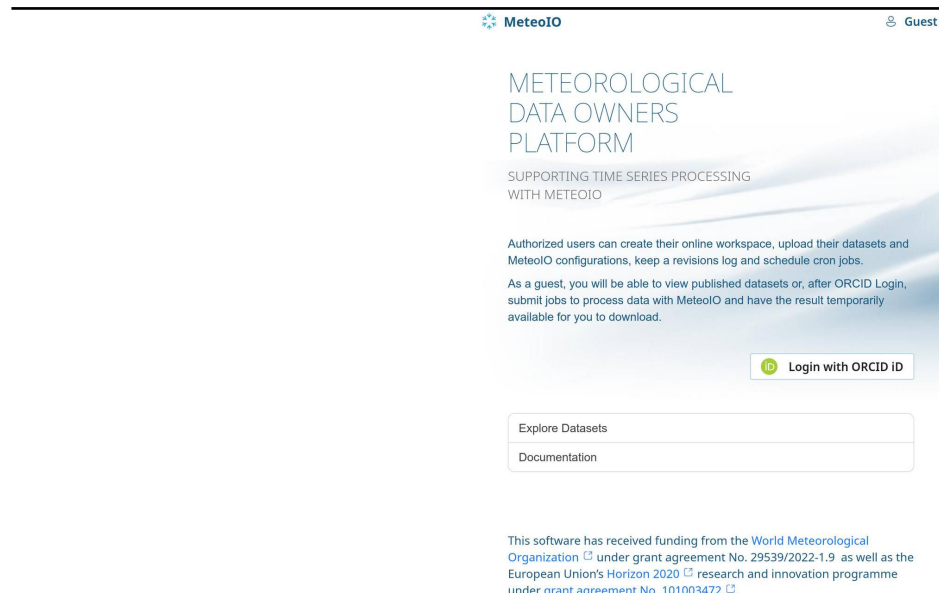


Figure 4. Screenshot of the landing page of the MeteoIO web service

These three different roles lead to different views and panels in the web service graphical user interface.

3.1. Anonymous data users

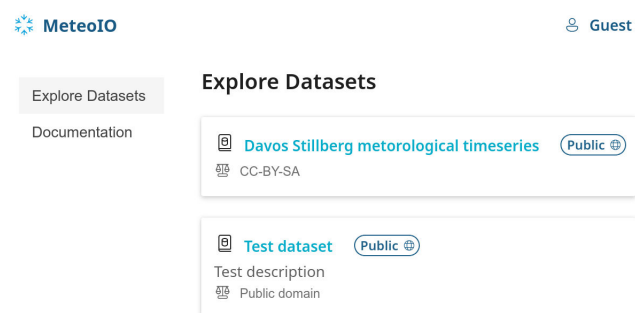



Figure 5. Screenshot of an anonymous user (not logged in) browsing public datasets

The login options are currently either by email and password (the account must then be created by an administrator, through the administration interface) or with an OrcID iD^[9]. Without any form of login, it is only possible to browse and download datasets provided by data owners (this is to prevent abuses of the system).

3.2. Guest users



User 74dddd77

Guest Job Submission

View last submissions

Working directory

STB_Orion.ini1.13 KB

STB_Orion_example_raw.csv279 B

+ Add files...

2 files, 1.41 KB

INI configuration

STB_Orion.ini

Select the INI file to configure MeteoIO

Range

1970-09-26T00:00:00

End

Current Time

...or insert a duration...or select an end date

Resolution


Value from INI configuration SAMPLING_RATE_MIN

... or specify a duration

Launch job

Figure 6. Screenshot of the submission of a guest job

3.3. Data owners



User 74dddd77

Output

Source Data

INI configuration

Cron

Logs

Settings

+ Add...

Folder

↑ /

↻

Name ↑	Date	Size
input	12 days ago	4.1 KB
acdd.ini		
STB_Orion_envidat.ini		

Figure 7. Screenshot of the configuration of a job by a data owner

The metadata that can be edited through the web interface is intentionally very limited: the vast majority must be provided in the ini file itself and only a few fields that are useful for the data owner (in order to easily know which job is related to which dataset) and for users browsing for data are exposed and all but one^[11] default to values extracted from the ini file.

Users who are logged in can still browse publicly available datasets but also submit guest processing jobs: they can manually upload one or more configuration files (ini files) and one or more data files, configure the start and end date of the dataset to generate and launch the standardization job (Figure 6). Once the processing has completed, they can browse the logs and download the resulting files. The processing usually takes a few seconds to complete (a few very long timeseries made of large number of files with complex parsing such as part of [Bavay & Fierz, 2023] take two to three minutes to be processed).

Users who have the *data owner* permissions can keep jobs permanently on the system. The upload of data files can be automated with the sftp protocol^[10] and the processing of the data can take place automatically at regular intervals as defined by the data owner. An extended configuration interface (Figure 7) allows to configure the data upload (either manually or through sftp), the automatic processing of the job (*cron* functionality), explore the logs generated by the processing and edit some of the metadata.

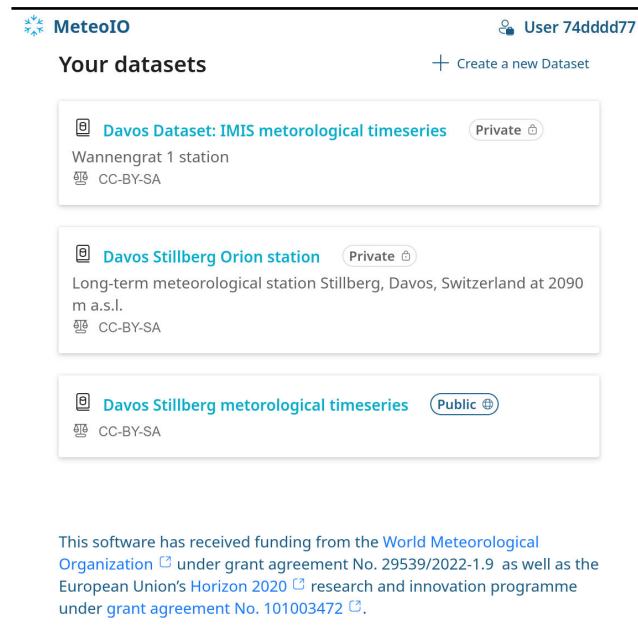


Figure 8. Screenshot of the configuration of a job by a data owner

metadata and data access for scientific datasets over a variety of protocols (data download over http, OpenDAP and others) [Parsons et al., 2011]. It is therefore possible to retrieve the data and metadata of public datasets at the following url: <https://service-meteoio.slf.ch/thredds>.

Data owners who have datasets on the system will see them on the home screen after logging in (Figure 8). They can then edit any of their existing datasets, rerun them as many time as necessary and then choose to manually publish the generated outputs in order to make them available to all users when they are satisfied with the results. This allows them to experiment with their datasets (for example in order to better filter some of the data) and only make them available to other users when they are ready. Please note that public datasets that are automatically generated at regular intervals (*cron* functionality) are automatically released and published when they succeed, superseding previous runs of the same *cron* job configuration entry.

A THREDDS server (Unidata's Thematic Real-time Environmental Distributed Data Services) is integrated into the webservice. It is a web server that provides

4. Technical implementation

4.1. Security considerations

One key issue that was identified early on in the project was the potential for abuse and hostile behavior on a system that lets users from the Internet run computing jobs on the server hosting the web service^[12]. Moreover, because at its core the system relies on the MeteoIO library and this library offers a lot of flexibility (as required by other projects using it), it can be dangerous from a computer security point of view. The most worrying features of MeteoIO for the current use case are the following:

- its ability to follow symbolic links;
- its ability to expand environment variables in its configuration file;
- its ability to create files anywhere on the filesystem (or at least attempt to, since normal filesystem permissions are enforced by the operating system);
- its ability to potentially generate very large files or very long running compute jobs (although this does not happen in practice, this could be a way to abuse the system, for example by requesting a decade-long dataset to be resampled to 100 Hz).

None of these capabilities can be removed as they might be essential for other use cases of MeteoIO (such as operational systems that rely on them in order to simplify their workflow). But they must be constrained in order to establish a layered defense against hostile behavior (and not solely trusting in the operating system to be properly configured to prevent damage).

The following set of measures have therefore been taken to mitigate the risks:

- MeteoIO can now be compiled with restrictions on file and directory creation, such as only allowing writing within the Current Working Directory (CWD);
- each job runs in a temporary, dedicated container, thus providing isolation from the filesystem of the server and between user jobs;
- guest jobs can only be submitted by users who have logged in through OrcID so abuses can be traced back;
- currently data owners must be manually vetted by an administrator (through a whitelist).

4.2. Docker containers

Docker containers are used for several different purposes: to enhance the security of the system by providing isolation between compute jobs, to integrate third party components and to make operational deployment easier (see [Figure 9](#) for the structure of docker containers).

The docker containers used for processing jobs in isolation are allocated from a pool of running containers when a client processing job is requested. Each contains MeteoIO and receives the necessary INI file and input files. After completing the processing, the container is destroyed while a new container is created in the pool. MeteoIO is built from source (pulled from its repository) on a Debian-slim base in order to create these images.

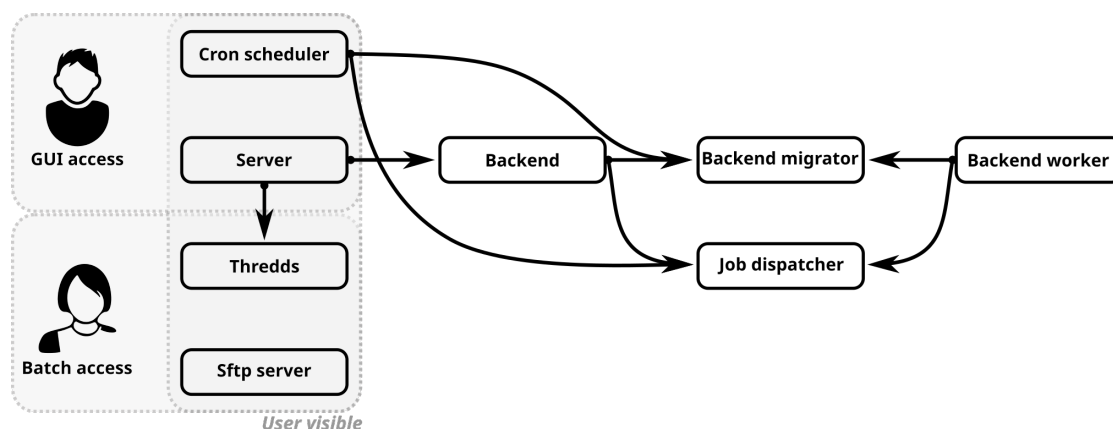


Figure 9. Overview of the docker structure

In order to be easy to deploy, the web service itself is also wrapped in a docker container: this reduces the software requirements on the server hosting the web service as well as the potential for configuration errors. This container is built by the server administrator when installing the system from source^[13]. It relies on docker compose to pull THREDDS and OpenDAP support from a third party (the Norwegian Meteorological Institute).

4.3. MeteoIO

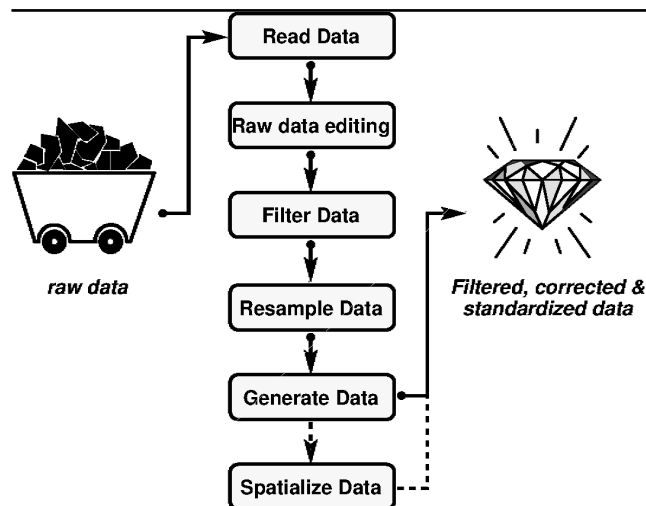


Figure 10. Overview of MeteoIO's workflow

by a configuration file in the INI format (within each processing step, nothing is hard-coded). The configuration is made through key/value pairs that are described in an extensive online documentation available at <https://meteoio.slf.ch>.

Reading and writing data is performed through the use of plugins^[15]. They are currently 12 plugins covering various file formats, web services^[16] and databases engines^[17]. The *input data editing* step consists of low level editing operations either at the parameter level (such as renaming parameters, swapping parameters (for example in the case of sensors connected to the wrong inputs of a data logger), deleting some parameters) or at the station level (such as merging stations together or editing the station's metadata). All these operations can be constrained by specific station ID and time ranges.

The data Quality Control (QC) is performed by the filters. These can either filter the data (removing individual data points deemed invalid) or correct the data (changing the value of individual data points). There are currently forty such algorithms that are applied on a per-parameter basis (including the timestamps) and can also be restricted by station ID and time ranges. These algorithms covers very generic methods (enforcing a valid data range, rate of change, high or low pass Infinite Impulse Response, Kalmann filtering), variance based filters (based on standard deviation, absolute median deviation, phase space outliers detection) or physically based algorithms (correction for sensor solar heating, detecting grass under a snow height sensor). Any number of such algorithms can be sequentially applied (building a filter stack) for each parameter. A special key allows MeteoIO to report every alteration applied to the data in its log file which then can for example be used to detect patterns of sensor failures.

Depending on the envisioned data quality level, it is possible to rely on the data resampling to deliver a dataset at a specific sampling rate as well as in order to perform gap filling. Six temporal resampling algorithms are available to choose from. If large gaps remain or if a parameter was not even available to start with, it is possible to rely on data generation algorithms. These range from very primitive (generating a constant value, sinusoidal wave) to quite performant parametrizations (generating incoming long wave radiation based on ground measured incoming short wave radiation^[18]) including lossless data transformations (generating a relative humidity based on measured dew point temperature and air temperature). Finally, the spatial interpolations can produce gridded data from point measurements or spatially interpolate point measurements to other coordinates with a choice of spatial interpolations algorithms (including Inverse Distance Weighting with detrending and kriging).

The MeteoIO meteorological pre-processing library has been in constant development since 2008 [Bavay & Egger, 2014]. It is distributed either in source code form, as pre-compiled binaries for various platforms or as a docker image^[14] on an Alpine Linux base image. MeteoIO is used for various operational applications (running unattended 24/7) as well as research applications (often embedded in numerical models). It is implemented in C++ which offers great computational performance and modularity. Various techniques are in use to make its operation robust, such as static code analysis and continuous integration. It relies on a fixed set of processing steps (Figure 10) where each processing step is fully defined

4.4. Inishell

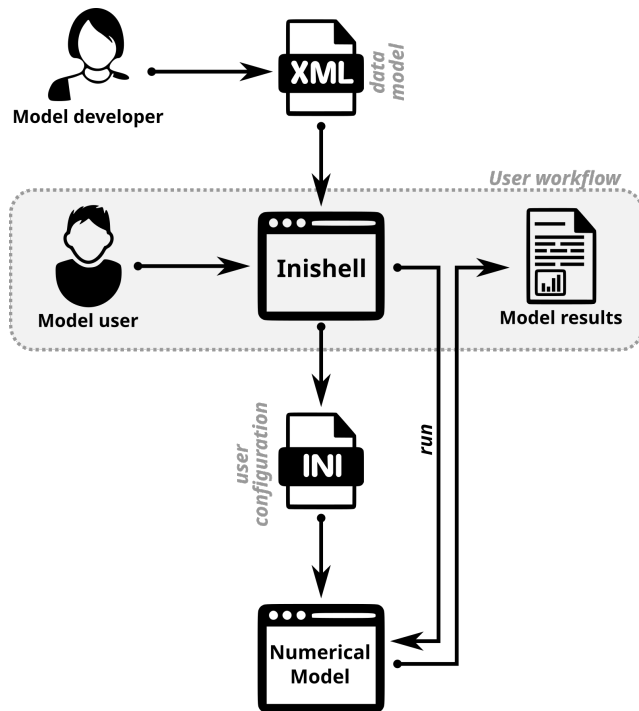


Figure 11. Principle of operation of Inishell

configure the underlying numerical model (in Arctic Passion, it is MeteoIO) and saves it as an ini file (Figure 11). The configuration keys are also most often provided with an help text that quickly summarizes how it behaves and often contains a direct link to the matching online documentation page for more detailed information.

When using the MeteoIO web service, the users are encouraged to rely on Inishell to write and edit their ini files before submitting them to the system.

5. Upcoming developments

Although the web service is already running and available for the first end users, it will still experience major developments in the next year. New features will be introduced and the developers will use the feedback from the end users to improve it. The work around MeteoIO and the MeteoIO webservice is currently funded thanks to several projects complementing Arctic Passion. One of these project, funded by the World Meteorological Organization, has provided the funds for the web GUI development that is visible to the end users of the MeteoIO webservice while the Arctic Passion funding has been focused on backend developments (on MeteoIO, Inishell and the pipelines to automatically publish releases and docker images on gitlab). The funding for upcoming developments that are discussed below has already been allocated from either Arctic Passion or from these other projects. This means that they should be implemented before the end of the Arctic Passion project, except if major technical difficulties were to arise.

There are several aspects of MeteoIO that will be further developed. New MeteoIO plugins will be written in order to add support for additional data sources or allowing export of data to additional platforms. This includes reading data from Copernicus' Wekeo data platform, reading and writing data in the GRIB and BUFR formats^[20] and reading data stored on Amazon S3 buckets. The temporal

Configuring numerical models^[19] is a major source of errors for users. These configurations are often highly technical, time consuming and require mastering a very large number of configuration parameters. Even when there is enough documentation, users tend not to read it therefore providing a Graphical User Interface is a major improvement to the efficiency of the users. But maintaining such as GUI when there is a very large number of configuration parameters that might change on short notice is a daunting task.

Inishell [Bavay et al., 2022] is an answer to this challenge that also proves useful for the Arctic Passion project. Instead of designing each entry panel with manually designed entry widgets for each configuration parameter, the configuration keys that will be exposed to the users are described in an XML file that is then used to dynamically build a GUI and show it to the user. The user then uses the GUI to

resampling will be enhanced by replacing the current single algorithm selection by a stack of resampling algorithms (similar to the current stack of filters) and implementing new resampling algorithm (such as ARIMA [Liston & Elder, 2006]). A system to attempt to automatically resolve dependencies between filters will be implemented (for example, if a filter on the air temperature requires radiation data, the radiation data should be filtered first). The filters that are applied on timestamps (as necessary to correct clock drift in a data logger for example) will be integrated into the data Quality Control logging. Support for profiles data (such as a water column temperature profile) will be implemented. A preliminary mechanism to propagate metadata from the input to the output will be implemented (instead on relying on the metadata to be provided in the INI file only). The possibility to define filters to be applied on gridded data (similarly to filters applied on timeseries) will be implemented.

The web service interface will also be further developed. The administration interface will be expanded in order to let the administrators define limits for the resources used by the compute jobs (currently such resources are set once and for all). The possibility for the users to request data owner permissions to the administrators through the system (instead of by email) will be implemented. Support for other OpenID^[21] providers will be added.

The data owner interface will support providing data file with an rsync daemon^[22] as well as providing data files on local network shares. A monitoring interface will be provided to graphically show the time distribution of data issues as logged by MeteoIO alongside an alert system based on MeteoIO logs (for example, sending an email to the data owner when a given sensor experiences too many rejected data points). The INI files have some primitive sort of versioning that will be enhanced in order to allow browsing arbitrary previous versions or showing the differences between two arbitrary versions. A prototype of interoperability with a data portal for final data publication will be build around the Envidat^[23] data portal. It will consist of populating the Envidat metadata fields from the metadata already present in the data files as well as offering DOI reservation to the data owners (so a DOI can be reserved on Envidat^[24] and written into the data files as metadata before finalizing the data publication on Envidat). At a later point, the web service will also be extended to expose MeteoIO's support for gridded data in the interface.

Finally, some training will also be provided to the users. At first, one-on-one sessions will be conducted over video calls in order to identify any typical issues that users might encounter and address them. This could lead to changes in the web interface, the back end or the Inishell GUI as well as changes in the documentation. The documentation will also be expanded (including with step by step examples) and enhanced by providing video tutorials (how to use the system for a simple dataset or how to setup filters in order to detect sensors problems and receive automatic notifications).

References

- [Bavay & Egger, 2014] Bavay, Mathias and Egger, Thomas. "MeteoIO 2.4.2: a preprocessing library for meteorological data", Geosci. Model Dev., 7 (2014), 3135–3151, <https://doi.org/10.5194/gmd-7-3135-2014>.
- [Bavay et al., 2020] Bavay, Mathias, Joel Fiddes, and Øystein Godøy. "Automatic Data Standardization for the Global Cryosphere Watch Data Portal." Data Science Journal 19 (2020): 6-6, <https://doi.org/10.5334/dsj-2020-006>.
- [Bavay et al., 2022] Bavay, Mathias, Reisecker, Mickael, Egger, Thomas, and Korhammer, Daniella. "Inishell 2.0: semantically driven automatic GUI generation for scientific models", Geosci. Model Dev.,

15 (2022), 365–378, <https://doi.org/10.5194/gmd-15-365-2022>.

- [Bavay & Fierz, 2023] Bavay, Mathias, Fierz, Charles. "The Davos Dataset: an (un)fair backstory", 28th International Union of Geodesy and Geophysics General Assembly, Berlin (2023), <https://doi.org/10.13140/RG.2.2.33863.68009>.
- [Brun et al., 2013] Brun, E, Lawrimore, J, de Rosnay, P and Friddell, J., "Proposal for a GCW action aiming at improving operational snow depth observation for snow depth", Technical Report, Global Cryosphere Watch (2013), URL: <https://globalcryospherewatch.org/projects/snowreporting.html>.
- [Jung et al., 2016] Jung, Thomas, et al., "Advancing polar prediction capabilities on daily to seasonal time scales", Bulletin of the American Meteorological Society, 97.9 (2016), 1631-1647, <https://doi.org/10.1175/BAMS-D-14-00246.1>
- [Liston & Elder, 2006] Liston, Glen E., Elder, Kelly, "A meteorological distribution system for high-resolution terrestrial modeling (MicroMet)", Journal of Hydrometeorology (2006), 7.2, 217-234, <https://doi.org/10.1175/JHM486.1>
- [Parsons et al., 2011] Parsons, MA, Godøy, Ø, LeDrew, E, De Bruin, TF, Danis, B, Tomlinson, S and Carlson, D, "A conceptual framework for managing very diverse data for complex, interdisciplinary science", Journal of Information Science, 37 (2011), 555–569, <https://doi.org/10.1177/0165551511412705>
- [de Rosnay et al., 2015] de Rosnay, P, Isaksen, L and Mohamed, D. 2015. "Snow data assimilation at ecmwf". ECMWF Newsletter, 26–31. URL: <https://www.ecmwf.int/en/elibrary/17328-snow-data-assimilation-ecmwf>, <https://doi.org/10.21957/lkpxq6x5>

[1] See <https://arcticpassion.eu/> for more details on the Arctic Passion project.

[2] Funding agencies require more and more often FAIR data management. But the (I)nteroperability and @eusability are often sub-par.

[3] A section of MeteIO's documentation is dedicated to developers who want to extend it, including by implementing new filtering algorithms. A test performed for [Bavay & Egger, 2014] showed that for an average user without C++ experience and only provided with the online documentation, it takes less than an hour to implement a simple filter.

[4] MeteIO is available at <https://code.wsl.ch/snow-models/meteoio>

[5] Inishell is available at <https://code.wsl.ch/snow-models/inishell>

[6] The MeteIO-webservice is available at <https://code.wsl.ch/snow-models/meteoio-webservice/>

[7] The THREDDS data server is available at <https://github.com/Unidata/tds>

[8] Please note that although the SLF must host one instance of the web service to make it available for the Arctic Passion project, the MeteIO webservice is open source and its installation is documented so it is envisioned that other providers will host their own instance at some point in time.

[9] For more information on Orcid, see <https://orcid.org/>

[10] The Secure File Transfer Protocol is an extension to the secure shell protocol version 2.0 of the Internet Engineering Task Force. Both the sftp and scp data transfer tools are now built on top of the sftp protocol.

[11] The only metadata field that is certainly not extracted from the ini file is the dataset visibility, which can either be *public* or *private*. This field is reserved for the web service and can be changed in the dataset properties.

[12] A first version of the meteio webservice was implemented in early summer 2022, tightly integrating the web service within MeteIO. Unfortunately, this could not provide sufficient security and therefore the approach that is presented here was chosen. An archive is kept at <https://code.wsl.ch/snow-models/meteoio/-/releases/webservice-2022> for historical purposes.

[13] The installation instructions for the MeteIO-webservice are provided in the source repository at <https://code.wsl.ch/snow-models/meteoio-webservice/>

[14] MeteIO docker images are available from https://code.wsl.ch/snow-models/meteoio/container_registry and are automatically generated when a new push is made to the repository (*latest* image) or when a new release is tagged in git (*release* image).

[15] Although the MeteIO plugins used to be compiled separately and dynamically linked, this is not the case anymore. In order to enable or disable a plugin, MeteIO must now be recompiled.

[16] MeteIO supports for example the Meteoblue API, www.meteoblue.com

[17] MeteIO has plugins for Postgresql and MySQL. The latter can be extended in a few minutes to accept new queries to get the location

metadata and the data. By design such queries must be declared in the source code, thus reducing the risk of SQL injection.

[18] The number of plugins, filters, parametrizations does not give a full picture of what MeteoIO has to offer to handle the data: for example, the incoming long wave parametrization (appearing as one such algorithm) actually consists of five different published parametrizations that the user can chose from, applicable from the Middle East deserts to Greenland.

[19] Examples of numerical models are the Weather Research & Forecasting Model or the SNOWPACK snow cover model

[20] There is already a plugin to read GRIB files, but it requires large scale restructuring. It is based on the ecCodes library provided by the European Centre for Medium-Range Weather Forecasts at <https://github.com/ecmwf/eccodes>.

[21] <https://openid.net/>

[22] <https://rsync.samba.org/>

[23] Envidat is the data portal of the WSL institute in Switzerland. It is available at <https://envidat.ch>

[24] DOI reservation on Envidat is explained at https://www.envidat.ch/#/blog/new_doi_workflow.md