natESM

Sprint Documentation #13

MESSy-Import server

Aleksandar Mitić[1], Astrid Kerkweg[2], Patrick Jöckel[3], Kerstin Hartung[3], Bastian Kern[3] , and Timo Kirfel[2]

[1] Deutsches Klimarechenzentrum GmbH, Hamburg, Germany
[2] Forschungszentrum Jülich GmbH, Jülich, Germany
[3] Deutsches Zentrum für Luft- und Raumfahrt (DLR) e.V., Oberpfaffenhofen, Germany

Contact: info@nat-esm.de

Published on 24.09.2026 on https://www.nat-esm.de/services/support-through-sprints/documentation

1 Summary

The MESSy[1]-Import-server sprint aimed to develop an asynchronous input[2] server to improve the model performance by isolating the input process from the computational tasks. The sprint focused on implementing an import component within the MPMD[3] setup of the modeling framework using the YAC[4] library. The IMPORT server was developed in Python, whereas MESSy is dominantly written in Fortran. The server reads input data in the highest available resolution and remaps it online to the required model grid.

The main challenges addressed during the sprint included a revision of YAC and MPI implementations in MESSy to support existing MPMD setups, an implementation of time management for asynchronous data reading across various fields with different time frequencies, an automatic detection of model grids from CF-compliant NetCDF files, a replacement of the current `IMPORT_GRID` submodel by `IMPORT_GRIDSRV`, and adapting the IMPORT server to the main model features and configurations.

The main outcome of the sprint was an asynchronous IMPORT server that leverages YAC's features, including asynchronous communication, regridding, and re-partitioning, enhancing the MESSy's overall model performance. All goals defined in the kick-off meeting were achieved with several additional features beyond the original scope, such as time series interpolation of the fields, CI tests in the GitLab repository, additional base models YAC grid implementations, and the successful integration of the automatic YAC grid-detection developments into the IMPORT server.

The sprint therefore not only improves the performance of current MESSy simulations, but also establishes a reusable infrastructure that can support future developments within the MESSy framework.

[1] Modular Earth Submodel System
[2] "import" in MESSy terms
[3] Multiple Program Multiple Data
[4] Yet Another Coupler

2 General Information

Start and end date:	October 2025 – May 2026
Intended period:	06 months
Responsible RSE:	Aleksandar Mitić, DKRZ
Responsible scientist:	Astrid Kerkweg, FZJ-ICE 3

The Modular Earth Submodel System (MESSy, Jöckel et al., 2005, 2010, 2016) is an integrated software framework that provides the infrastructure (data types and methods) for coupling atmospheric models (e.g., ICON, ECHAM, COSMO) with specialized Earth System Model (ESM) components, the so-called submodels (e.g., physical parameterizations, chemistry packages, diagnostics) via generalized interfaces for standardized control and coupling [1]. The code repository is currently hosted on the DKRZ GitLab server, and work towards an open-source release is ongoing.

The YAC Coupler Library (Yet Another Coupler) is a high-performance, modular software framework for coupling Earth system model components. It provides robust support for grid interpolation, parallel data exchange, and time coordination between heterogeneous model components. YAC is built for scalability and flexibility, enabling accurate and efficient coupling on massively parallel HPC systems [2].

3 Sprint Objectives

When operated as a chemistry-climate (and air quality, AQ) model, MESSy requires a large amount of gridded input data. In the previous implementation, importing these fields was tightly coupled to the computational tasks, so that I/O operations were performed synchronously during model execution. Depending on the host model, this led either to input data being read by dedicated processes and subsequently distributed to compute processes, as in EMAC, or to all processes accessing the same input files concurrently, as in DWARF, COSMO, and ICON. Both approaches introduced an I/O bottleneck during the simulation. To overcome this limitation, an additional dedicated MPI process was introduced to substitute the existing IMPORT_GRID submodel. The new IMPORT server is coupled online to the new MESSy submodel IMPORT_GRIDSRV, i.e., the main model and communicates the read data asynchronously utilizing the YAC library. The implementation also required a multi-node performance evaluation and comprehensive documentation of its usage.

4 Procedure and Insights

4.1 Technical Approach / Procedure

Revision of the MPI and YAC implementation in MESSy: In order to support existing MPMD setups with YAC usage and the existing YAC based Output server, MPI initializations were reworked. This required using a version v3.14.0_p of the YAC library (at the time of the report v3.19 exists and is used in MESSy), for which synchronization points for information exchange between selected components, either MESSy and IMPORT or MESSy and the Output server, were needed. Flexibility of the YAC implementation was taken into account, as a user can also compile the code with or without YAC enabled, and still have the option to perform the runs w/o the I/O servers. Additional adaptation of run scripts for both, Input and Output server, were implemented so that the existing MESSy's MMD (multi model driver) script setups were not disrupted. At present, the support status of the MESSy base models with the IMPORT Server configuration is summarized below:

- DWARF [3]: The IMPORT server can run with multiple DWARF instances and dedicated MPI tasks will be assigned to each of the base model components.
- EMAC: The AGCM setup (atmospheric general circulation model without interactive chemistry) was used in addition to ICON for the performance analysis of the IMPORT server as EMAC featured a different interpolator (NREGRID) implementation than ICON (SCRIP).

- ICON: The IMPORT server works for one (global) ICON domain. Extension of this setup to multiple domains requires several changes: (A) the YAML configuration file needs to be extended by an input file specific optional choice of target domain(s), which needs to be handled in the IMPORT server and on the MESSy side, respectively. (B) Multiple target grids (as many as domains) need to be supported. The support for nested domains was omitted in this sprint, as supporting the nested-domain configuration introduces an additional layer of implementation complexity that could not be addressed within the available project time.

  During the sprint, further effort was required to maintain compatibility between the ICON and MESSy models, as both code bases continued to evolve and adopted newer YAC features.

- The COSMO base model support is in place, but never actually tested. The MECOn setup (MESSy-ECHAM5 + two COSMO instances with MMD library) requires a rework of YAC implementations inside MESSy due to synchronization points between the parent and the child models, and the corresponding I/O component servers where the call tree is not in sync between all of the components.

During the sprint we also developed a set of scripts to install tailor-made python environments, which comprise the netCDF bindings for python to be consistent with the underlying HPC installation of the MPI library and the netCDF library, as well as the required YAC and YAXT(Yet Another eXchange Tool) libraries and bindings on the Levante HPC system.

Setting up the Import component in Python: A script was carefully designed to follow MESSy's call tree which reads the input files and its configuration file, that replaces existing Fortran namelists by YAML formatted input. The IMPORT server reads multiple netCDF files, where for each it automatically defines a YAC grid and a conservative remapping to MESSy's model grid. The automatic grid detection algorithm from netCDF files, which are CF Metadata Convention compliant, was developed by MESSy team members and was merged to the IMPORT server parent branch. All the fields are read and to each of them a `start_date` is assigned based on the time difference of its netCDF time bounds representation and the simulation start (or restart) date. The end date is calculated based on the field's frequency, which can be represented as ISO 8601 duration string. Additionally the users can still override this calculation and adjust the start, end time, and data cycling indices in the YAML configuration file, as it was in the original namelist setup. From this point, an additional feature was implemented for linear time series interpolation (4.1) between two data points.

$$d_1(t_{x2}) = d_1(t_{x1})\frac{\Delta t - \Delta t_y}{\Delta t} + d_1(t_{x3})\frac{\Delta t_y}{\Delta t} \tag{4.1}$$

, where a linear interpolation is performed between two adjacent imported data points $d_1(t_{x1})$ $d_1(t_{x3})$. The interpolated value $d_1(t_{x2})$ is calculated as a weighted average, where the weights depend on the relative position of t_{x2} within the interval $[t_{x1}, t_{x3}]$, t_y is elapsed time from the earlier data point to the interpolation time, and $x1, x2$ and $x3$ are labels identifying different time instances.

As the formula requires two data points to be present already at the starting point of the time loop - t_0 , the IMPORT server sends two time steps already in the beginning asynchronously to the MESSy component which can be seen in Figure 1. The time series feature is configurable by the user for each individual import file.

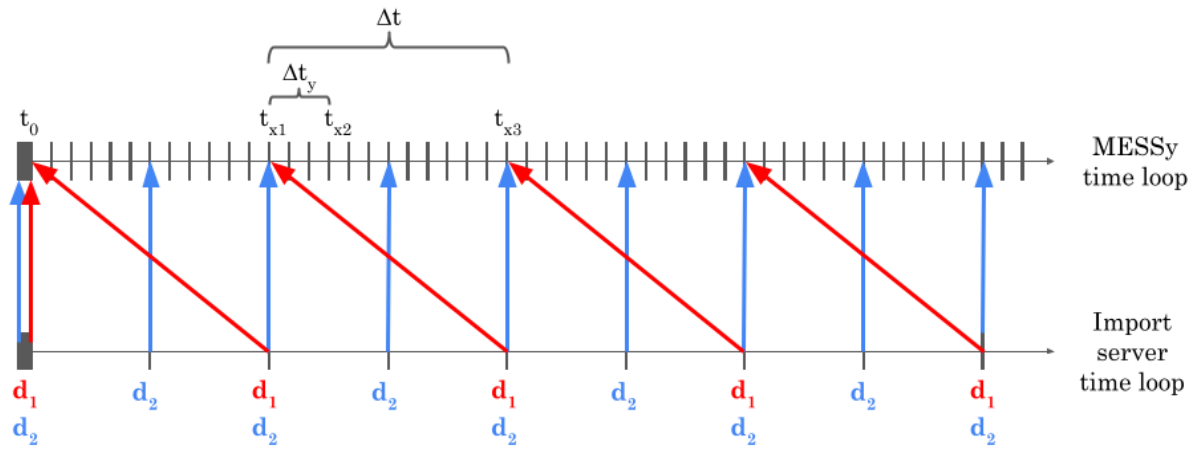

Figure 1: Example of two communication patterns from the IMPORT server to the MESSy models during time loop: for **d1** (red, time interpolation enabled) field, each imported netCDF time slice is used across the interval until the next slice and interpolated to the model times; for **d2** (blue, interpolation disabled) field, values are passed directly at their discrete file times.

If the user configures an import event with time interpolation enabled, then the original simulation end date calculation is extended in case the data point from the netCDF file is in the future.

The **natESM software testing strategy** is to employ testing infrastructure into the code, therefore an adapted CI test and a set of unit tests were implemented. From testing the python script, to the build stage of the model, and making a dependency between jobs for the final stage of running the DWARF with both I/O servers enabled was developed.

4.2 General Insights

Having a dedicated I/O process is meant to have reduced impact on model runtime, with simulation time improvements. Making the communication asynchronous between the components provides better overlap of computation and data handling and it also reduces the blocking of compute ranks. As these I/O servers are python based, this approach increases portability of the model, and gives users a flexible interface development. Therefore, a legacy MPI setup must be adapted to include these I/O ranks.

Although YAC is primarily designed as a coupling interface rather than an I/O library, this sprint demonstrated that it can also serve as an effective mechanism for runtime input provision by linking independent Python-based preprocessing workflows with MESSy. In the presented framework, temporal interpolation is carried out on the target grid, whereas YAC performs the horizontal interpolation and repartitioning to the model target grid. This design minimizes data transfer, reduces MPI message overhead and memory usage, and enables a modular and scalable workflow [4]. Because YAC computes its horizontal interpolation weights during the setup phase rather than amortizing this cost across the time loop as NREGRID and SCRIP do, it also contributes to the substantial reduction in initialization time observed in the Import server configuration, alongside the time loop savings shown in the Results section.

5 Results

The newly developed IMPORT server and MPMD configuration were subject to performance analysis and profiling on the DKRZ HPC system Levante and its compute nodes. The simulation was based on a production setup and compiled with the Intel compiler version available at the time of the experiments. Profiling of the EMAC MPMD setup was carried out using the open-source tool Score-P, and the resulting performance data were analyzed with the Cube framework [5,6]. EMAC's model code could be instrumented directly, which made this more extensive, Score-P-based analysis practical.

Within the MESSy time-loop profile, execution times obtained with the standard input routines were compared against those of the new input server process. **EMAC** was configured at T42L90MA resolution (128 × 64 × 90 grid boxes), and the global atmospheric general circulation model configuration (EMAC/AGCM) using ~50 netCDF input files was

selected for the analysis. The simulated period was 4 months, with a time step length of 5 minutes and monthly and yearly input frequencies. Since the input server reads the time bounds of the fields, fields are not necessarily sent on the first day of each month, but instead according to their actual representative time.

A SLURM batch script was developed to support MPMD execution on Levante, specifically for scenarios in which an I/O server task runs concurrently with the climate model. Performance was evaluated for three resource-allocation strategies: multithreading, in which the server shares a hardware thread on the model nodes; hetjob, in which the server is assigned a dedicated node for full isolation; and pack distribution, in which the server runs on a separate physical core.

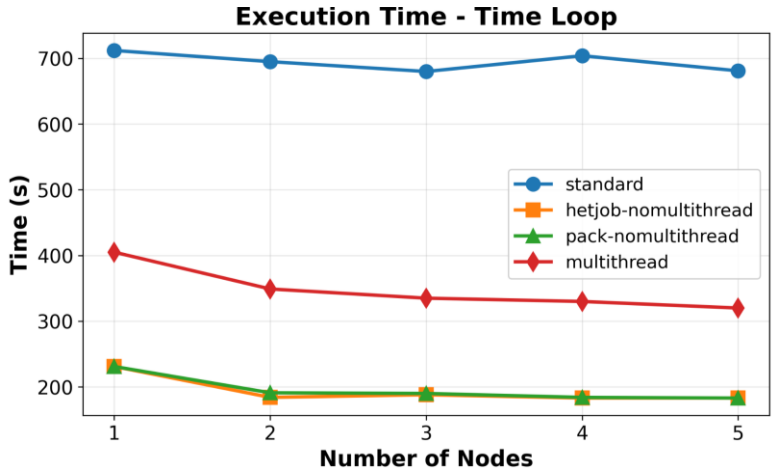

Figure 2: Models time-loop timing across 1-5 Levante compute nodes for four configurations: the **standard** MESSy simulation is consistently the slowest; **hetjob** and **pack** are the fastest and show very similar scaling despite using an additional N+1 server node as the horizontal (XY) domain decomposition of ECHAM5 as the base model of EMAC is limited in task separation along X and Y (both limits depend on the resolution) and therefore prohibits prime numbers of total tasks in most of the practical applications; and the **multithreaded** setup - running the server on the same N nodes as the model but sharing the resources with one compute task.

In Figure 2, a speedup of factor of ~3.5 can be seen when compared between standard and non-multithreaded approaches. This gain results from asynchronous data delivery to MESSy, which minimizes blocking in the compute processes and improves overall throughput. Based on the observed speedup and the reduced blocking of compute processes, the pack-nomultithread configuration is recommended for production simulations.

The same IMPORT server approach was additionally tested with **ICON**, using a comparable setup of ~50 netCDF import files. Since the goal of these runs was to characterize scaling behavior rather than to reproduce a full production integration, the simulated period was reduced to 1 month, in contrast to the 4 month EMAC experiments, but still enough to see multiple import trigger events. Unlike the EMAC/Score-P analysis, profiling of the ICON setup did not use Score-P, since doing so would have required modifying the precompiled ICON code to support instrumentation, a comparatively time consuming step. Instead, simple built-in MPI wall clock timers were used, which proved sufficient to characterize the relative performance of the standard and IMPORT server configurations.

In contrast to EMAC, where in the old standard NREGRID is used, importing and regridding on one task and scattering the result, ICON/MESSy uses SCRIP, importing and regridding on each task for the respective local domain. The parallel read access was one reason for the encountered bottleneck with SCRIP. Using SCRIP, the weights are calculated in the initialisation phase of the simulation. As this is computationally the most expensive part, the sprint application identified the initialization time as one of the bottlenecks. Therefore, an additional Figure 3 showing the initialization timings is included alongside the time loop comparison in Figure 4. Task distribution for ICON also differed from the EMAC case: since ICON was run with an odd number of total processes, the server task could be placed on the same node as the model.

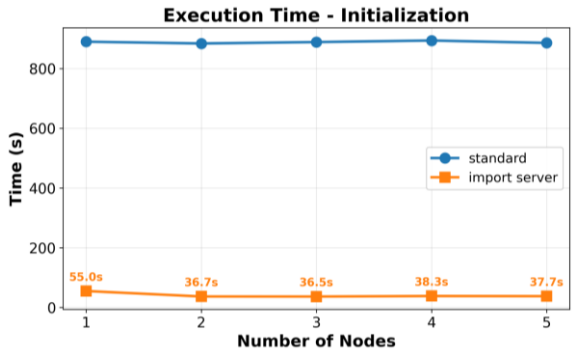

Figure 3

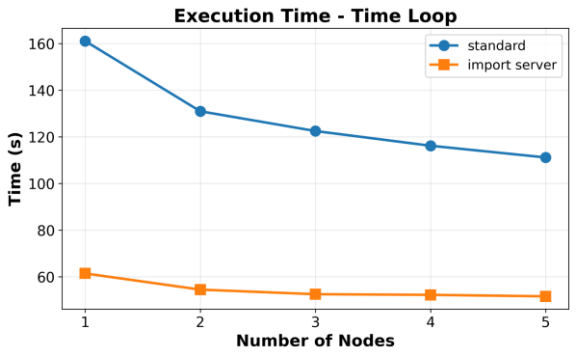

Figure 4

Overall, the time loop with the IMPORT server is approximately 2.3× faster on average than the standard ICON-MESSy import, confirming that offloading import handling to a dedicated server process substantially reduces per-step overhead in addition to the initialization gains discussed above.

Part of the remaining timing differences, however, is not purely architectural: the standard ECHAM5/MESSy import path relies on NREGRID for interpolation on one task followed by scattering of the input data. ICON/MESSy's remapping uses SCRIP on each task for its respective local domain, which slows down the simulation due to all tasks accessing the same file for reading at the same time. Finally, the IMPORT server performs its interpolation online through YAC, so the three configurations are not only distinguished by their I/O server architecture but also by distinct interpolation algorithms and implementations, which likely contribute to part of the observed timing spread.

6 Conclusions and Outlook

This work introduced and evaluated a newly developed IMPORT server for the MESSy framework on the DKRZ Levante system. Performance analysis showed that the new approach can significantly improve runtime behavior compared with the standard input routines, primarily because asynchronous data delivery reduces blocking in the compute processes. The results further indicate that the achievable performance gain depends on the selected resource-allocation strategy and execution configuration. Overall, the presented implementation demonstrates the potential of decoupling input operations from the main model workflow to improve portability and efficiency in climate-model applications.

Future work should focus on extending both the performance and the functionality of the implementation. In particular, its scalability toward exascale systems should be investigated, accompanied by further optimization of input strategies. In addition, the IMPORT servers could be parallelized more extensively by leveraging YAC-based capabilities, such as domain decomposition and shared-memory techniques. To be able to run the MECOn system with all IMPORT servers for all base model instances, the YAC communication needs to be reworked establishing independent YAC instances for the import and the output server. Another important extension is the integration of support of the ICON configurations featuring nested-domains. In terms of functionality, some features, e.g., index regridding, need still to be added to the server before the server can completely replace IMPORT_GRID. Finally, increasing robustness with respect to different data-reading patterns and input formats would further enhance the general applicability of the approach.

7 References

[0] Code documentation can be found on: gitlab.dkrz.de/MESSy/MESSy/-/merge_requests/1552

[1] Jöckel, P., Sander, R., Kerkweg, A., Tost, H., and Lelieveld, J.: Technical Note: The Modular Earth Submodel System (MESSy) - a new approach towards Earth System Modeling, Atmos. Chem. Phys., 5, 433–444, https://doi.org/10.5194/acp-5-433-2005, 2005.

[2] Hanke, M., and Redler, R.: New features with YAC 1.5.0, https://doi.org/10.5676/DWD_pub/nwv/icon_003, 2019.

[3] Kerkweg, A., Kirfel, T., Do, D. H., Griessbach, S., Jöckel, P., and Taraborrelli, D.: The MESSy DWARF (based on MESSy v2.55.2), Geosci. Model Dev., 18, 1265–1286, https://doi.org/10.5194/gmd-18-1265-2025, 2025.

[4] Redler, R., Dreier, N.-A., Schneidereit, A., Haak, H., and Hanke, M.: ICON input via Python, https://doi.org/10.5676/DWD_pub/nwv/icon_012, 2026.

[5] C. Roessel, VI-HPS :: Projects :: Score-P. [Online]. Available: https://www.vi-hps.org/projects/score-p/ (accessed: 12-Jun-26)

[6] P. Saviankou, M. Knobloch, A. Visser, and B. Mohr, "Cube v4: From Performance Report Explorer to Performance Analysis Tool," Procedia Computer Science, vol. 51, pp. 1343–1352, 2015, doi:10.1016/j.procs.2015.05.320.