Sprint Documentation #15

MESSy ComIn 2

Lakshmi Aparna Devulapalli[1], Aleksandar Mitic[1], Bastian Kern[2], Kerstin Hartung[2], Patrick Jöckel[2], Astrid Kerkweg[3]

[1] Deutsches Klimarechenzentrum GmbH, Hamburg, Germany
[2] Deutsches Zentrum für Luft- und Raumfahrt, Institut für Physik der Atmosphäre, Oberpfaffenhofen, Germany
[3] Forschungszentrum Jülich GmbH, Institute of Climate and Energy systems, Troposphere (ICE-3), Jülich, Germany

Contact: info@nat-esm.de

Published on 25.07.2025 on https://www.nat-esm.de/services/accepted-sprints

1 Summary

The goal of this sprint was to continue the progress achieved in natESM sprint #08 - MESSy-ComIn. The broad goal for both sprints was to develop the Modular Earth Submodel System (MESSy) as a first complex plugin for the ICON Community Interface - ComIn, which acts as an interface between ICON-A and external plugins. Prior to the two sprints, MESSy was integrated with ICON through hard-coded entry points. The idea is to remove the hard-coded components of the MESSy/ICON model, and make MESSy the first complex plugin for ICON through ComIn.

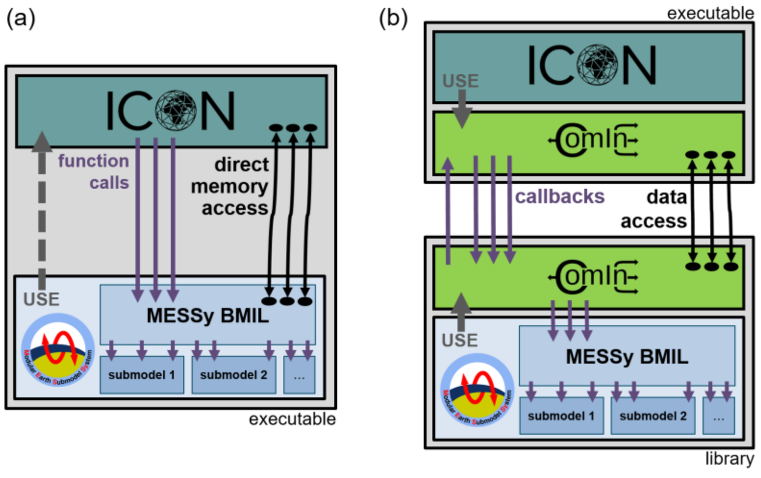

Figure 1: natESM sprints goal [1]

At the start of the sprint, the responsible Research Software Engineer (RSE) had just joined the natESM team at DKRZ. Accordingly, the first weeks were dedicated to onboarding, getting familiar with internal workflows, and understanding the developments made during sprint #08 (4).

Throughout the sprint, there was active collaboration amongst the natESM RSE, representatives of the MESSy-developer group and the ComIn-developer group. The sprint also benefited from close coordination with another RSE based at DKRZ and funded through the ESiWACE project, who worked on related developments in parallel. His contributions are marked with "(EW)" in this report.

As part of the sprint, contributions can be recorded in all three relevant software packages: MESSy, ComIn, and ICON. At the end, all the functionalities were successfully tested with the provided test setups before handing the work back to the scientists for finalization of the development.

In this sprint report, the reader will find information about the major tasks undertaken as part of the sprint, technical details associated with each task and other relevant information.

2 General Information

Start and end date:	September 2024 – March 2025
Intended period:	06 months
Responsible RSE:	Lakshmi Aparna Devulapalli, DKRZ
Responsible scientist:	Bastian Kern, DLR-PA

Below is a brief introduction to the software packages used in the sprint, including links to their code repositories.

MESSy https://gitlab.dkrz.de/MESSy/MESSy : MESSy is a software framework that combines components, which are numerical representations of processes in our Earth system. Examples of components are physical and chemical processes in the atmosphere, and their interaction with land and ocean, and more.

ComIn https://gitlab.dkrz.de/icon-comin/comin : The Community Interface (ComIn) organizes the data exchange and simulation events between the ICON model and "3rd party modules". The concept can be logically divided into an Adapter Library and a Callback Register.

> Adapter Library: It is included in both the ICON model and the 3rd party module. It contains descriptive data structures, and regulates the access to existing and the creation of additional model variables.
>
> Callback Register: Subroutines of the 3rd party module may be called at predefined events during the model simulation.

ICON https://gitlab.dkrz.de/icon/icon : ICON is a flexible, scalable, high-performance modelling framework for weather, climate and environmental prediction that provides actionable information for society and advances our understanding of the Earth's climate system.

ICON is evolving into an internationally recognized and widely used modeling framework for weather, climate, and environmental prediction that allows the users to solve challenging problems of high societal relevance. It will push the boundaries of what is computationally feasible to advance knowledge and understanding, and to support innovation in weather, climate and environmental services.

Setup for sprint development: To build a working setup, firstly, ICON is built by enabling ComIn as an external library. Secondly, MESSy is configured using ComIn and ICON as targets. The entire simulation is driven using an ICON generalized run-script, starting the ICON/ComIn model, which in turn loads the MESSy shared library and executes its main entry points via registered callbacks. This experiment setup was used throughout the sprint to test developments.

Since this sprint is a continuation of a previous natESM sprint (see natESM sprint report #08 [4]), a previous MR was merged into the `devel` branch (the main development branch of MESSy) before the beginning of the sprint. MESSy contribution standards were followed during branches and issue creation inside the MESSy gitlab repository. During the entirety of the sprint, two big MRs were created to merge the work back into the `devel` branch.

3 Sprint Objectives

The sprint objectives were divided into the following major tasks:

1. Automatic channel-object creation: A standardized algorithm was required for channel-object creation for all the ICON fields which are being exposed via ComIn. ComIn provides access to all the ICON fields through a convenient linked list, and the idea is to loop through them to execute necessary actions on the MESSy side. This includes creation and association of appropriate dimensions and representations with the channel object created for each field.
2. Dimensional Semantics: This task was added in the sprint objectives subsequently, as it became clear during the sprint that this was a critical requirement for the success of task 1. The goal was to improve the metadata of the source data produced by ICON, by adding information on the order and meaning of dimensions.
3. YAXT communication patterns - Gather, Scatter, Boundary Exchange: MESSy is a highly parallel software. In its prior state of hard-coded integration with ICON, MESSy was using communication methods provided by ICON. But with the goal of detachment, dedicated communication methods were needed to be developed for MESSy.
4. MPI communication – one-to-one and all-to-all communication: MESSy depends on routines provided by ICON to do MPI communication. In the spirit of further detaching MESSy from ICON, a new MESSy-specific interface for MPI communication needed to be developed.
5. ComIn callback calls for entry points in time loop - ComIn callback calls needed extension to cover the MESSy entry points in the ICON time loop. This involved converting some local fields into global fields through patches for MESSy's operation.
6. Resolving dependencies - There is a huge list of miscellaneous dependencies on ICON, which are majorly highlighted by the modules `mo_mpi` and `bmluse`. These dependencies needed to be addressed to achieve the final goal of complete detachment from ICON.

4 Procedure and Insights

4.1 Technical Approach / Procedure

Automatic creation of Channel Objects: The primary objective of this task was to create channel objects in a generic way replacing the existing error-prone algorithm, which was a long list of "if statements". This included more universal naming nomenclature for dimensions and representation.

The creation of channel objects is a multistep process. [2] Two key ingredients for the creation of channel objects are the underlying dimensions and representations. `Dimension` information is stored in a data structure called `t_dimension`, and each dimension is identified uniquely by its name and id, which are stored in this data structure. `Representation` is a MESSy concept, which describes the underlying geometric structure of the data. This information is also stored in a data structure `t_representation`, which holds many meta-data and attributes related to the geometric layout of the fields on a grid. And similar to dimension, a representation is also uniquely identified by its name and id. Hence, to create a channel object for a field, we need to provide the corresponding dimension and representation information, either by creating new ones or by using existing ones from the list of dimensions and representations, respectively. The following approach was adapted to achieve this objective.

The idea is to loop over all the fields being exposed by ICON/ComIn, and to perform the following actions during each loop iteration:

1. **Create generic names for dimension and representation** based on the geometric information provided by ICON/ComIn. For example: (i) `DIMID_NCELLS` for the horizontal dimension holding the information of number of cells (ii) `LEV90` for vertical dimension which constitutes of 90 vertical levels (iii) `CELL2D` - representation which can hold a 2D field on cell centers (iv) `EDGE3D_XZY_LEV91` - representation which can hold a 3D field stored on the edges of the grid, "`XZY`" indicating that the order of the three dimensions and "`LEV91`" representing the number of vertical levels.
2. **Check if the dimension name already exists**; if not, create a new dimension using the generated dimension name.
3. **Check if the corresponding representation exists**; if not, create a new representation, based on the existing or newly created dimensions in the previous step.
4. **Use dimension id and representation id** from previous steps to create a new channel object.

Some of the default dimensions and standard representations are created separately for specific cases in ICON/MESSy.

The above approach depends on accurate geometric information received from ICON/ComIn. This brings us to the curious case of missing dimensional semantics.

Dimensional Semantics: The ICON/ComIn interface provides a pointer to a 5D pointer, which is associated with the memory holding the data. Upon deeper inspection of the dimensions and dimension lengths of these data, we found out that not all fields have a standard ordering of the dimensions. To understand which dimensions represent the horizontal, vertical, or other dimension was critical for generating the representations as per the nomenclature decided above.

ComIn interprets each 5D pointer and heuristically determines the position indices for all dimensions. This heuristic breaks for the fields defined in ICON, which do not follow the standard convention of dimensional ordering. To solve this problem, and have complete reliable information from ICON, "dimensional semantics" was proposed as an additional meta-data for variables. This idea was presented during the ICON all hands meeting in October 2024 and was received positively by the participants of the respective breakout-group discussion.

The implementation of "dimensional semantics" is still in progress as we submit this report. This task was divided into further subtasks:

1. **Interface Modification in ComIn to receive dimensional semantics**: As part of this step, we removed the position indices and replaced them with dimensional semantics. ComIn internally declares the list of named constants in the same way as ICON, to assign the semantics to each dimension in the form of an additional array named `dim_semantics`.
2. **Adapting MESSy code to utilize dimensional semantics provided by ComIn**: Now, the automatic channel- object- generation algorithm uses dimensional semantics to check for the

vertical dimension and then captures the size of that dimension to generate the name of representations containing vertical dimensions.

3. **Implementing the changes in ICON** (MR in progress): A separate module was created to support the changes being introduced by dimensional semantics. This module contains the named integer constants and functions for sanitation checks. The add_var() subroutine was extended to have an optional argument `dim_semantics` that will eventually become a mandatory argument once all the add_var() calls in ICON source code are modified to provide the dimensional semantics.
4. **Removing the heuristic from ComIn** (To be done): The final step is to clean up the heuristic from ComIn files to only receive the final dimensional semantic information without needing to modify it further.

YAXT Communication Patterns: [3] YAXT (Yet Another eXchange Tool) was chosen as the library to create one-to-all and all-to-one communication methods in MESSy for data transpositions. MESSy currently performs a `gather` operation to write output using one MPI process and performs a `scatter` operation after every restart to read from the restart files using one MPI process and distribute the data to all the remaining processes.

In YAXT terminology, a communication method is called a YAXT redistribution. This redistribution object of type `yaxt_redist` was set for all the representations. MESSy would then be able to perform gather and scatter transformations for each field based on the redistributions assigned to each of them through representations.

Since both underlying models, ICON and MESSy, commonly are based on a 2D (horizontal X-Y domain decomposition), the implementation for 2-dimensional grid-point related data was straightforward, whereas 3D and 4D grid-related data was handled differently. 3D and 4D fields required calculation of the offsets of local indices to pack 3- and 4-dimensional data onto a 2-dimensional point set. This required critical understanding of how memory is organized in ICON to develop the logic for YAXT.

MPI – all-to-all communication (EW): While heading towards further detachment of the hard-coded ICON dependencies in MESSy, which for this case was all-to-all communication, an entirely new ICON-independent MPI interface was implemented for MESSy. It contains a MESSy-only MPI module for interfaced communication routines for point-to-point and collective communications as well as asynchronous communication. As a result of this implementation, MESSy now relies on the workgroup communicators provided by the Base Models (ICON, ECHAM, COSMO, etc..), while using its own internal routines for MPI communication. In the case of ICON/ComIn/MESSy, the work communicator is provided via a ComIn function.

Calls inside time loop (EW): For the already developed ComIn plug-in, MESSy entry points inside the time loop had to be added via callback functions for ComIn. The full functionality of the implemented callbacks is still under development, but the implementation of the routines didn't cause any change of results, as tested on Levante. MESSy now accesses additional entry points both, before and after ATM_MICROPHYSICS, ATM_TURBULENCE, ATM_GWDRAG and ATM_PHYSICS.

4.2 General Insights

Getting familiar with the large scientific code bases of MESSy and ICON was the first major task before the sprint even began. The initial month of onboarding at DKRZ in the natESM project group was spent getting to know the processes, handling complex GitLab repositories, and understanding the prior work done as part of the first MESSy-ComIn sprint (see 4).

The natESM workshop on coupling with YAC and ComIn interface conducted in July 2024 turned out to be the perfect opportunity for an in-person meeting between the scientists and the RSE. This fostered a great bond at the start of the sprint for a successful collaboration in the coming months. Responsible scientists and the RSE got additional opportunities to meet in person in the form of workshops, conferences and hackathons over the course of the sprint. This further improved collaboration and communication within the sprint team.

All the technical decisions were taken post extensive discussions during the weekly sprint meetings. An interesting aspect of the sprint was how the design strategies of MESSy and ComIn influenced each other. The core team of MESSy developers were also part of the core team of ComIn developers, and this promoted synergy between the development of both software packages.

Most of the code developments in MESSy happened in the base model interface layer (BMIL) in the ICON specific file. This led to some design decisions being taken to follow the MESSy coding conventions. This means, the RSE was suggested to use public data access functions already available as part of MESSy, instead of directly accessing the private data structures from the internal lists. This strategic decision ensured clean control flow between different modules of MESSy.

One of the other significant insights obtained during the preparational work for "automatic channel object creation" task in MESSy, is that there is no reliable way to know the order of the dimensions in the way ICON fields are defined. This discovery led to a vital improvement in the ICON code base, which is essential for ComIn as well to translate reliable and accurate data to its plugins.

Developing YAXT-communication methods for MESSy presented an interesting insight into how fields are stored in the memory. This knowledge was critical to calculate offsets for higher than 2 dimensional fields, and for sending data across processes in the right order.

Throughout the sprint, we encountered a bunch of technical issues, but thanks to the support of the scientists and other RSEs we were able to resolve them and continue making progress. A continuous challenge was to keep the MESSy code up-to-date with the changes being introduced in a dynamically developing ComIn interface. Once ComIn reaches its version 1.0.0, our expectation is that there will be no major changes in the interface for a while.

5 Results

A robust "automatic channel object creation" algorithm was developed for ICON/ComIn/MESSy. The list of fields was accumulated and the algorithm was checked thoroughly for all the unique cases that may occur. Significant time was spent understanding the geometric definition of the fields to create corresponding dimension variables and representations for the fields. A functionality test was successfully conducted to test the working of the implemented algorithm.

Groundwork was laid for introducing dimensional semantics in ICON. Since this was a major change in ICON source code, it was divided into multiple steps and the final implementation in ICON is expected to finish in some time. But MESSy and ComIn have already been adapted to receive the future changes coming from ICON.

YAXT-based communication methods for gather and scatter operations were developed in MESSy to remove the dependency on ICON's communication subroutines and functions. As a result, the YAXT redistribution objects are now associated with representations, and gather and scatter operations can be performed using the information from the representation data structure. Tests were conducted to match the values after using YAXT for data transpositions with the reference solution using ICON's communication patterns. In the end, all the values were identical to the reference solution. Due to time constraints, YAXT- communication patterns could not be developed for boundary-exchange operations in MESSy.

The joint work with a Research Software Engineer from the ESiWACE project, who was working on related MESSy developments in parallel, contributed to several key outcomes of this sprint. These included the development of a new modular MPI interface in MESSy, which streamlined the use of communication calls and aligned with the broader goal of detaching MESSy from hard-coded ICON structures. The collaboration also helped resolve remaining dependencies on ICON. In addition, the team investigated the ComIn callback infrastructure required for integrating MESSy into ICON's time loop. Basic function calls were implemented in MESSy's ComIn plugin module to support this, though further adaptations are needed to access certain local fields required by MESSy.

In the end, we could not achieve all the goals decided at the beginning of the sprint, but foundational work was implemented for the scientists to develop it further. The following two tasks were handed over to the scientists for further development. YAXT communication patterns for boundary exchange and adapting ICON block loop variables to access via ComIn.

The sprint ended at the end of March 2025, and as of 04 July 2025, the MESSy team achieved a milestone in their journey of complete detachment from the hard-coded ICON. All the dependencies on hard-coded ICON have been resolved, and MESSy can now be built as a completely independent (without hard-coded ICON) shared library in the ICON/ComIn/MESSy setup.

6 Conclusions and Outlook

Through this second MESSy-ComIn sprint, we were able to achieve further steps towards the goal of detaching MESSy from ICON, and developing it as the first complex plugin for ComIn. The entire sprint was a joint project executed by the RSE and scientists working simultaneously on various tasks to achieve the goal. As the RSE was dedicatedly working on the major sprint objectives, scientists supported in providing conceptual clarity and resolving bugs, which could only be resolved by having in-depth knowledge of the MESSy software. A successful collaboration with the ComIn team was a byproduct of this sprint. ComIn developments influenced the work being done for MESSy, and some of the design strategies of MESSy influenced decisions for ComIn.

At this stage, the following steps represent a likely path forward for ComIn/MESSy:

1. The responsibility for finishing the remaining two subtasks for the complete implementation of dimensional semantics in ICON lies with the RSE, and she hopes to achieve this soon.
2. All ICON dependencies in MESSy have been resolved and scientific evaluation of the new ICON/ComIn/MESSy setup, which is independent of hard-coded ICON, is yet to be performed. Work in this direction is in progress by the scientists, and they hope to achieve complete removal of hard-coded ICON after successful testing.
3. MESSy and ComIn continue to be closely associated in the future, and once MESSy is completely operational only through ComIn, MESSy will become a benchmark for all the future complex plugins to be developed with ComIn as an interface.

7 References

1. **Figure 1:** https://messy-interface.org/about/national-earth-system-modelling/
2. **MESSy CHANNEL User Manual for the MESSy CHANNEL submodel**. This manual is part of the electronic supplement of our article "Development Cycle 2 of the Modular Earth Submodel System (MESSy2)" in Geosci. Model Dev. (2010), available at: geoscientific-model-development.net
3. **YAXT documentation** : https://dkrz-sw.gitlab-pages.dkrz.de/yaxt/
4. **Sprint report #08:** https://www.nat-esm.de/services/support-through-sprints/accepted-sprints/240429_sprint_report_messy_comin.pdf

8 GitLab repositories

1. **MESSy repository:** https://gitlab.dkrz.de/MESSy/MESSy
2. **ComIn repository:** https://gitlab.dkrz.de/icon-comin/comin
3. **ICON repository:** https://gitlab.dkrz.de/icon/icon