Technical Memo

ECMWF
European Centre for Medium-Range Weather Forecasts

904

Software Strategy and Roadmap 2023–2027

Tiago Quintino, Umberto Modigliani, Florian Pappenberger, Sylvie Lamy-Thépaut, Simon Smart, James Hawkes, Iain Russell, Laurent Gougeon, Cristiano Zanna, Baudouin Raoult

January 2023

Series: ECMWF Technical Memoranda

A full list of ECMWF Publications can be found on our website under:
http://www.ecmwf.int/en/publications

Contact: library@ecmwf.int

European Centre for Medium-Range Weather Forecasts, Shinfield Park, Reading, RG2 9AX, UK

Abstract

ECMWF develops several software packages to support the primary purpose of the Centre, to develop a capability for medium-range weather forecasting and the provision of medium-range weather forecasts to the Member States, as well as the objectives aligned with this purpose[1]. This strategy is in support of the "ECMWF Strategy 2021–2030"[2] in particular the pillar on *Science and Technology* which aims to develop and make operational use of cutting-edge science and technology and the pillar on *Impact* which aims to provide exceptional value for money to ECMWF's Member States. Therefore, many of the software packages are also used by Member and Co-operating States as well as other users to process and analyse ECMWF archived or disseminated data. These software packages have been developed over many years, sometimes decades, and continue to provide a reliable service to ECMWF operations.

As part of our constant drive for efficiency and improvement, ECMWF is reviewing its strategy and roadmap for these software packages. The general principles that underpin ECMWF's software strategy are as follows:

- **Efficient use of development resources**. We aim to continue supporting, with the available development resources, our current services, whilst providing capability to continue to grow both in breadth and depth. To realise this goal, we need to rationalise and be efficient in all development activities, as well as to minimise software maintenance efforts. This will be achieved by modernisation, reducing technical debt, and consolidating functionality so that it incurs less maintenance cost.
- **Software reusability.** To consolidate the development efforts, we will merge common functionality into reusable components with well-defined interfaces that encourage usage in a variety of environments, from HPC workloads to interactive end-user sessions.
- **Software componentisation and integration.** To enhance reusability, existing software will be refactored into smaller units of functionality. We aim to make these components smaller, more reusable, and simpler to integrate with each other. We also recognise that some of our software does not integrate well with Python, a common environment for our users and the community in general. We want to make these components more integrated with Python.
- **Further use of external software.** We aim to strike a good balance between our own development of domain-specific software that is critical to the delivery of ECMWF's core mission, with the use of well-maintained and supported community software. This will leverage the community interaction to further strengthen our developments whilst allowing us to focus on the functionality that is our core mission.
- **Adoption of open development.** We believe that software developed openly with interaction and feedback from the community leads to increased quality. Our experience has shown that the cost-benefit is very favourable and leads to enhanced interaction with Member and Co-operating States.
- **Data scalability.** One of the drivers of this software strategy is the required improvements in scalability of data handling, as the forecast data sets grow more than quadratically, with increases in resolution and new scientific parameters. We aim to refactor our software stack to improve its use of data in particular stages of the workflow, to minimise expensive data access operations, such as

[1] Article 2 of the "Convention Establishing the European Centre for medium-range Weather Forecasts"

[2] https://www.ecmwf.int/en/about/what-we-do/strategy

network or I/O operations; and where possible provide the software tools and services to support operating on the data where it is, i.e., a more data-centric workflow.

- **Modernisation to new standards for higher interoperability.** Over the years some new standards have emerged, particularly in the landscape of web technologies, that bring higher levels of interoperability between systems within and outside our data centre. This increased interoperability improves access to ECMWF data and is strongly aligned with efforts such as the move to open data and adoption of FAIR principles.

Plain Language Summary

This document outlines how ECMWF approaches the software side of the organisation (software strategy) and the short- and long-term solutions to achieve this approach (software roadmap). We plan to improve parts of the ECMWF collection of independent software components that work together to support the ECMWF forecast systems, in particular the data handling, post-processing, and service to users. We present the general principles that underpin the software strategy, and a roadmap which addresses ten areas of intervention, including four horizontal areas which impact many operational systems. Software related to the development of the IFS, verification and diagnostics is considered out of scope, although they do utilise software described herein.

Table of Contents

1. Software Strategy

ECMWF develops several software packages to support the primary purpose of the Centre, to develop a capability for medium-range weather forecasting and the provision of medium-range weather forecasts to the Member States, as well as the objectives aligned with this purpose[3]. This strategy is in support of the "ECMWF Strategy 2021–2030"[4] in particular the pillar on *Science and Technology* which aims to develop and make operational use of cutting-edge science and technology and the pillar on *Impact* which aims to provide exceptional value for money to ECMWF's Member States. Therefore, many of the software packages are also used by Member and Co-operating States as well as other users to process and analyse ECMWF archived or disseminated data. These software packages have been developed over many years, sometimes decades, and continue to provide a reliable service to ECMWF operations.

As part of our constant drive for efficiency and improvement, ECMWF is reviewing its strategy and roadmap for these software packages. The general principles that underpin ECMWF's software strategy are as follows:

- **Efficient use of development resources**. We aim to continue supporting, with the available development resources, our current services, whilst providing capability to continue to grow both in breadth and depth. To realise this goal, we need to rationalise and be efficient in all development activities, as well as to minimise software maintenance efforts. This will be achieved by modernisation, reducing technical debt, and consolidating functionality so that it incurs less maintenance cost.
- **Software reusability.** To consolidate the development efforts, we will merge common functionality into reusable components with well-defined interfaces that encourage usage in a variety of environments, from HPC workloads to interactive end-user sessions.
- **Software componentisation and integration.** To enhance reusability, existing software will be refactored into smaller units of functionality. We aim to make these components smaller, more reusable, and simpler to integrate with each other. We also recognise that some of our software does not integrate well with Python, a common environment for our users and the community in general. We want to make these components more integrated with Python.
- **Further use of external software.** We aim to strike a good balance between our own development of domain-specific software that is critical to the delivery of ECMWF's core mission, with the use of well-maintained and supported community software. This will leverage the community interaction to further strengthen our developments whilst allowing us to focus on the functionality that is our core mission.
- **Adoption of open development.** We believe that software developed openly with interaction and feedback from the community leads to increased quality. Our experience has shown that the cost-benefit is very favourable and leads to enhanced interaction with Member and Co-operating States.
- **Data scalability.** One of the drivers of this software strategy is the required improvements in scalability of data handling, as the forecast data sets grow more than quadratically, with increases in resolution and new scientific parameters. We aim to refactor our software stack to improve its use of data in particular stages of the workflow, to minimise expensive data access operations, such as

[3] Article 2 of the "Convention Establishing the European Centre for medium-range Weather Forecasts"

[4] https://www.ecmwf.int/en/about/what-we-do/strategy

network or I/O operations; and where possible provide the software tools and services to support operating on the data where it is, i.e., a more data-centric workflow.

- **Modernisation to new standards for higher interoperability.** Over the years some new standards have emerged, particularly in the landscape of web technologies, that bring higher levels of interoperability between systems within and outside our data centre. This increased interoperability improves access to ECMWF data and is strongly aligned with efforts such as the move to open data and adoption of FAIR principles.

1.1. Resourcing

The general principles of our software strategy are designed to increase efficient use of resources, through seeking synergies in component reuse, use of open-source software and enhanced collaboration with Member and Co-operating States.

Over the last decade, the software development at ECMWF has benefited significantly from external funding for strategy-relevant deliverables. Today, only 1/3 of the development resources are supported by Member and Co-operating State funding, with 2/3 obtained from a mix of external sources such as Copernicus, Destination Earth, other EU research programmes and other national funding sources. This document provides a single software strategy, irrespective of the funding source, as these programmes are drivers to developments that directly contribute and benefit the core mission of ECMWF.

ECMWF will continue to seek resources in a similar manner, complemented by enhanced collaborations with Member and Co-operating States, or the Centre of Excellence created by the Atos HPC contract; whilst relying on outsourcing of activities when economically and technically viable.

1.2. Innovation

A key motivation for seeking collaboration through the aforementioned channels, such as Copernicus, Destination Earth or EU research programmes, is to drive innovation in our software. These collaborations synergize with the core mission of ECMWF, helping us to prepare for higher resolution forecasting, utilization of new technology or exploring new avenues of development. These projects also allow us to work closely with a diverse range of partners across Europe, which keeps us abreast of cutting-edge research.

EU research programmes have given us the opportunity to work with state-of-the-art I/O hardware and develop novel data storage backends for the FDB (Fields Data Base), improving our readiness for new hardware. Other research programmes have allowed us to develop novel algorithms for data-cube feature extraction (Polytope), which will improve access to our data, particularly as our forecast resolution increases.

Important development areas featured in this document have strong synergies with these programmes, e.g., the reuse of web framework components from CADS to ecCharts, or MultIO developed for DestinE Digital Twins reused within IFS.

ECMWF's Innovation Platform is another key contributor to our innovation strategy, the aim of which is to offer a platform to facilitate collaborations and knowledge sharing, as well as providing the necessary data, software, and computing infrastructure to experiment with new ideas. Currently, the Innovation Platform is focusing on the creation of tools to facilitate the development of AI/ML models on the Centre's data.

Additionally, ECMWF runs the *Code for Earth* as an innovation programme. Each summer, up to ten external developer teams work together with experienced mentors from ECMWF and Copernicus on open-source projects related to software development, web development, machine learning or applied data science.

The aim of *Code for Earth* is to drive innovation and open-source developments in the Earth science community. ECMWF also hosts hackathons targeting certain themes, such as "Hackathon 2022: Visualising Meteorological Data", which bring fresh ideas from a wider community – not just developers.

Our strategy of increasing software reusability and software componentisation gives us more flexibility to explore new avenues of development and integrate novel solutions into our software stack.

1.3. Open Development

ECMWF has had an open-source policy for all non-IFS software for many years and has also been making much of its open-source software available on GitHub. However, in most cases our developers primarily interact with the code on our internal BitBucket servers, with the GitHub copy available for external users to use and contribute to. For the most part, Continuous Integration (CI) and Continuous Deployment (CD), issue handling and documentation are handled using our internal Atlassian systems.

Whilst we are aware that there is potentially the extra burden of managing the interactions with external contributors, experience in recent years has shown that the cost-benefit is very favourable, with the community quite often improving the code and, moreover, increasing the interaction with Member and Co-operating States (e.g., with the Met Office in odc and eckit, and Météo-France within Atlas library).

We aim to build on the progress already made in the further adoption of GitHub and other open platforms, including for CI/CD and software documentation. This will bring ECMWF's software practices into line with much of the rest of the scientific software community and encourage greater collaboration.

1.4. Security

Software security is consistently addressed throughout the development process and is implemented as part of a robust software development life cycle (SDLC) that includes security-specific phases and activities. ECMWF follows and adopts best practices and guidelines, from organisations such as NIST and OWASP, into the development process to help ensure that software security is consistently considered. In this software strategy we in particular focus on the following areas:

- Repository security - all software developments are versioned and traceable to the individual author in the Git repositories hosted internally in our own data centre. With the adoption of Open Development, some of these will be moved to cloud-based Enterprise services, such as GitHub, for which we will implement advanced security features integrated with our authentication and authorization systems, such as 2 factor authentication. This offers full traceability, immutability, and reproducibility of the software artifacts.
- Data security - most external data flows into and out of the data centre are secure using standard industry protocols. However, it is important to highlight that data security also covers the data used in the process of developing the software and services, which maybe static auxiliary data (masks, country shapefiles, etc) or test data (e.g., reference results). The storage and provision of this data needs to be secured and its traceability ensured. This area will require planned evolution as datasets grow and current methods become outdated.
- External facing services security - multiple services are exposed to the outside of the data centre. Although we have invested in the past in actively scanning these services for security flows by employing specialised third-party contractors, we believe more tests and assurances need to be put in place to insure a high level of security. This will involve both the services that interactively serve our

users but also the API endpoints of our automated systems. These tests would preferably cover multiple techniques, from penetration testing and code scanning to social engineering approaches.

- Code security - finally security also comes down to developing code that is secure by design. To ensure this we employ multiple approaches. Code committed is always reviewed before reaching operational status, either by peers or team leaders. With respect to open development, this means for external contributions only running CI/CD workflows on our infrastructure after careful code review from internal software maintainers (called Gatekeepers). We will also continue to employ automatic code quality tools and sanitizers that ensure that common mistakes like buffer overflows are less likely to occur. We will continue to have our code regularly scanned by specialised third-party contractors that look for code that is suspicious or marked as unsafe, or that infringes licenses or patents. Finally, we will invest in the training of developers to ensure that security is a particular concern when designing and implementing software and services.

1.5. Software Development Principles and Practices

Given the complex and numerous systems, services, and software packages that we maintain with reduced resources, developer productivity is a main concern.

ECMWF has long embraced multiple techniques derived from Agile methodologies, to improve productivity. Conscious that one size rarely fits all, we adapt these methodologies for each service and software stack. Depending on the situation, we make use of multiple agile techniques like code reviews, sprints, hackathons, SCRUM-style meetings, code refactoring, pair programming, etc.

As an example, one of the main blockers to developer productivity is code readability and maintainability. Industry studies suggest that code is more read than written by factors ranging from 7/1 up to 200/1. We improve code readability by making it a main criterion for code acceptance, and by encouraging development in teams, avoiding silos and single-author code bases. We do this by code reviews, pair-programming, and show & tell sessions.

One of our main tenets is of high-quality, domain-driven development. This means that we rely on being focused on our domain, to develop software that is highly optimized for ECMWF's mission and the support of our Member States.

Ultimately, our approach can maybe be best described as based on Lean Software Development principles[5] for the development process, coupled to the production environment via an adapted DevOps methodology.

2. Software Roadmap

In this section, the software roadmap for the coming five years is presented, detailing the action plan for adopting changes to our software stack to prepare it for our future challenges, in line with the software strategy.

ECMWF operates multiple workflows that involve multiple software packages. Some of these workflows are operational and time-critical; others are operational but not time-critical; some are research oriented; others co-shared with external entities such as national meteorological services (NMS). To facilitate the description and scoping of the roadmap, we will take the operational time-critical workflow as an example, but the

[5] https://en.wikipedia.org/wiki/Agile_software_development

roadmap applies to software in all these workflows. Figure 1 illustrates the ECMWF's time-critical workflow, from the acquisition of observational data into the data centre, all the way to the dissemination of the forecast products. Observational data is acquired by the *ecPDS* service from a multitude of data sources world-wide, and then passed to the *SAPP* system that processes and prepares them, eventually transforming them to the *ODB* form that is ready for ingestion by the model. The IFS, here understood as the aggregate of all its components that make up the Earth System model and data assimilation system, will assimilate these observations and generate the forecast output, delegating handling of the output to the *MultIO* component. MultIO may do further processing, and eventually encode the data into GRIB form, before storing it in the *FDB*. From there, and within minutes of the fields being produced, the *PGen* system will fetch them to produce user-defined products for the National Meteorological Services in the Member and Cooperating States and other licensed users. In parallel to PGen, the *Post-Processing* will also compute derived products from the model output, storing them in FDB.

All products are disseminated by ecPDS to NMSs and other licensed users. The forecast output will also be archived for posterity in the *MARS* system, with some non-structured data archived to the *ECFS* system. Both systems are later used by our users and researchers, to access data that eventually becomes off-line. Web services (e.g., *ecCharts*, *WebMars*, etc.) are also used to provide forecasters and users access to our forecast products in convenient, interactive ways, from within or from outside the data centre.

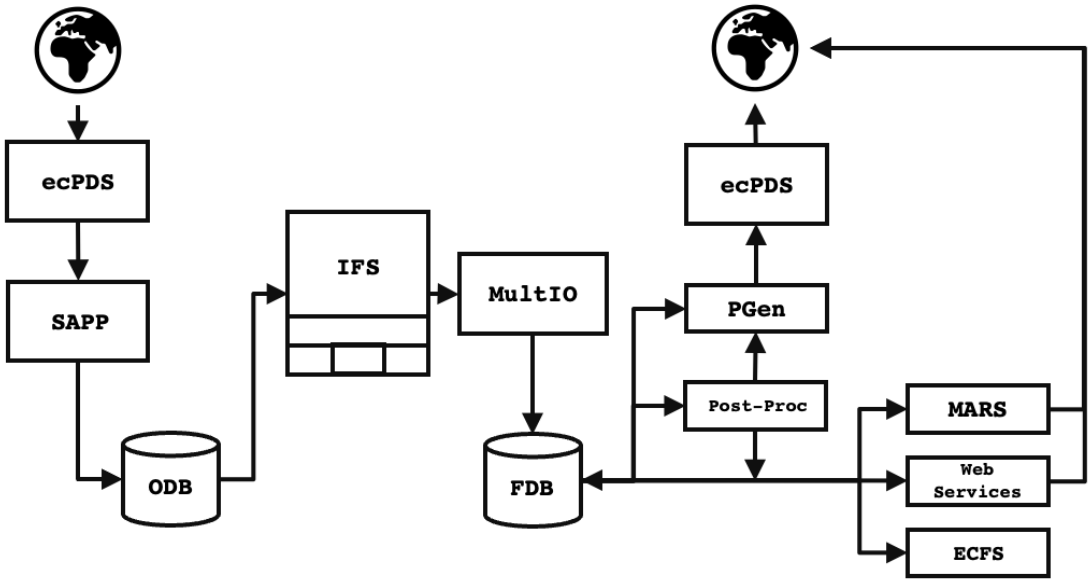


Figure 1: Generalised ECMWF workflow and data flow for generation of operational weather forecasts

The strategy described in Section 11 will be applied across the full depth of the software stack, and it will impact many of ECMWF's systems and services. To give an example, a single component like the GRIB data encoder/decoder (*ecCodes*), is used within the IFS model (via a Fortran interface), within the MARS archive (C/C++), inside the web stack, used in notebooks on the European Weather Cloud (Python) and ultimately provided to our NMS to process disseminated data (as a library and command line tools). This component of our software stack will go through a planned set of evolutions in the years to come, and therefore the benefits should be felt across the board.

2.1. Areas of Intervention

We present the roadmap by describing multiple areas of intervention, some in vertical areas related to specific applications, some in horizontal areas that crosscut along multiple applications and services to support them.

The areas of intervention foreseen for the next five years are outlined below:

- **Refactoring of the software stack** is a horizontal cross-cutting area, interacting with multiple applications and services.
- **Data encoding and decoding libraries** provide low-level functionality across the software stack to encode and decode ECMWF observations and forecast data. These libraries form a horizontal, supportive layer for all ECMWF services.
- **Visualisation software** is used across multiple applications and services, to display observation and forecast data, in forms familiar to the users and Member State forecasters. This is considered a supportive horizontal intervention area.
- **IO-server and on-the-fly processing pipelines** is a horizontal area, providing supporting libraries that give the Earth System Models the capabilities to asynchronously process and output their model output data, minimising the runtime of the time-critical runs by hiding I/O and allowing computations to proceed.
- **Data dissemination and acquisition** is a vertical area which provides the service that sits at the edge of ECMWF's data centre, managing the acquisition of incoming observations and the dissemination of the forecast products to NMSs and Users.
- **Observational data pre-processing** is a vertical area which provides the service to process the incoming observations, filter them, and ensure a common format before being fed into the Data Assimilation system.
- **Core data storage software** is a vertical area, providing the services that store, serve, and manage the data assets of ECMWF. This includes the managed and unstructured archives in the DHS system as well as within the HPC facility storage.
- **Post-processing framework** is a vertical area, delivering time-critical applications that create derived products from ECMWF forecasts, making them available to other services, and ultimately, the users.
- **Metview environment** provides an interactive, as well as batch, execution context for users to explore, process and visualise ECMWF data. This is considered a vertical area.
- **Web framework** is a vertical area, supportive of a myriad of web applications that serve users in multiple purposes at ECMWF, including data discovery and download.

2.2. Refactoring of the Software Stack

Status

Over the years, ECMWF has built a deep software stack with multiple layers to structure all the services and applications that compose our forecast production workflows and support users handling the forecast data. This stack, depicted in Figure 1, is mostly based on compiled software, often aimed at performance and scalability, with much of it making the backbone of our systems.

New software components such as *Atlas*, *MIR*, *ODC*, *PGen* or *FDB5* have been introduced to ensure that we remain at the forefront of data handling, scaling to ever larger and denser datasets. Other packages, such as

Metview, *Magics and ecFlow*, have been actively maintained to support new functionalities and ensure the best user experience. As the stack has grown, synergies have been sought to bring about savings in the maintenance cost of software and consolidation of functionalities. The strategy pertaining to this *compiled* software stack is spread over multiple areas of intervention, mostly described in subsequent sections.

Growing organically in parallel to this compiled stack, there is another stack based on the Python language. Originally this was mostly limited to the provisioning of language bindings to the existing C/C++ APIs. This led to a series of interfaces that were not very easy to integrate with other Python community packages. As more systems rely on the Python stack, it has become evident that ECMWF needs a more holistic approach to the Python developments, one where both developments in the compiled C/C++ stack and the Python stack are developed together, choosing the right language and interfaces for each task.

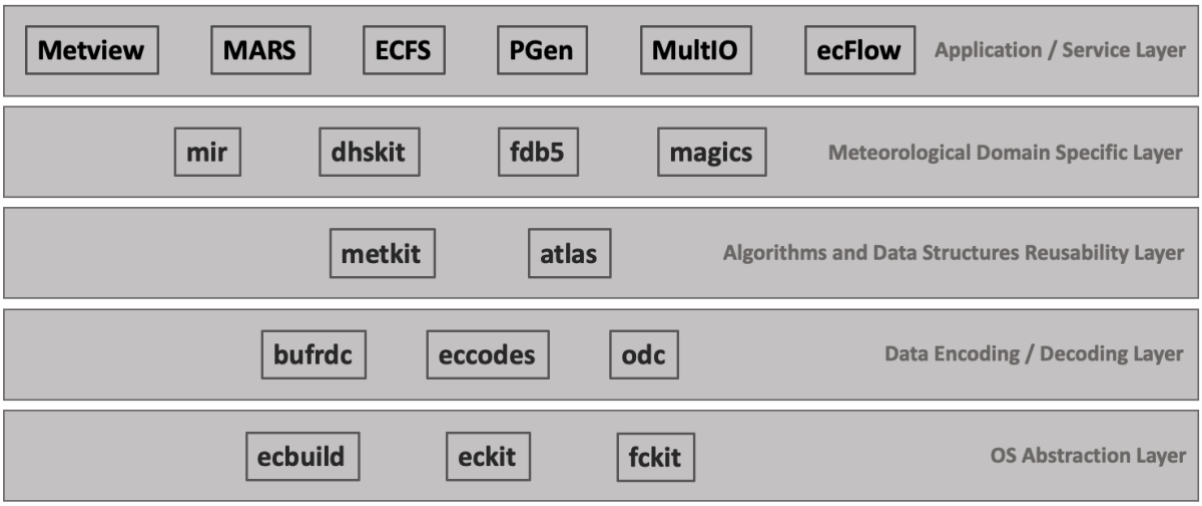


Figure 2: Current compiled software stack at ECMWF. Excludes Python components, web services and ECPDS (whose stacks aren't compiled). Some complex systems, with multiple components (e.g. MARS, ECFS) have been coalesced into a single component for simplicity.

Motivation

To make the software stack more reusable under a multitude of different execution environments, from massive parallel HPC workflows to web services, from interactive user Jupyter-like sessions, to batch data analytics, we have started a project named SPICE. This project will take a holistic approach to the refactoring of the software stack, making it more compatible with Python whilst also looking for rationalisation of functionalities, that were often duplicated, into dedicated packages for reusability. This project strongly relates to the future componentisation of Metview and the new MARS client. The aim is to create a set of components with well-defined interfaces that are reusable and composable, which can be used to engineer the more complex high-level applications and services that ECMWF maintains.

In Figure 3 we demonstrate what the future stack may look like. This view is centred around the packages that will be supporting the Python language, which supplements the compiled stack presented in Figure *2*. Many of these components will interface to the compiled software stack where functionality profits from delegation to a lower-level, high-performance C/C++ implementation. This is the case for the *mir-python, pyfdb, pyodc, eccodes-python* components that are shallow interfaces around compiled libraries. Over the next five years, we plan to refactor both software stacks simultaneously, as the needs of ECMWF applications and services arise. Large projects such as the new design of the CADS (formerly known as CDS), and the IFSHub developments are driving these developments.

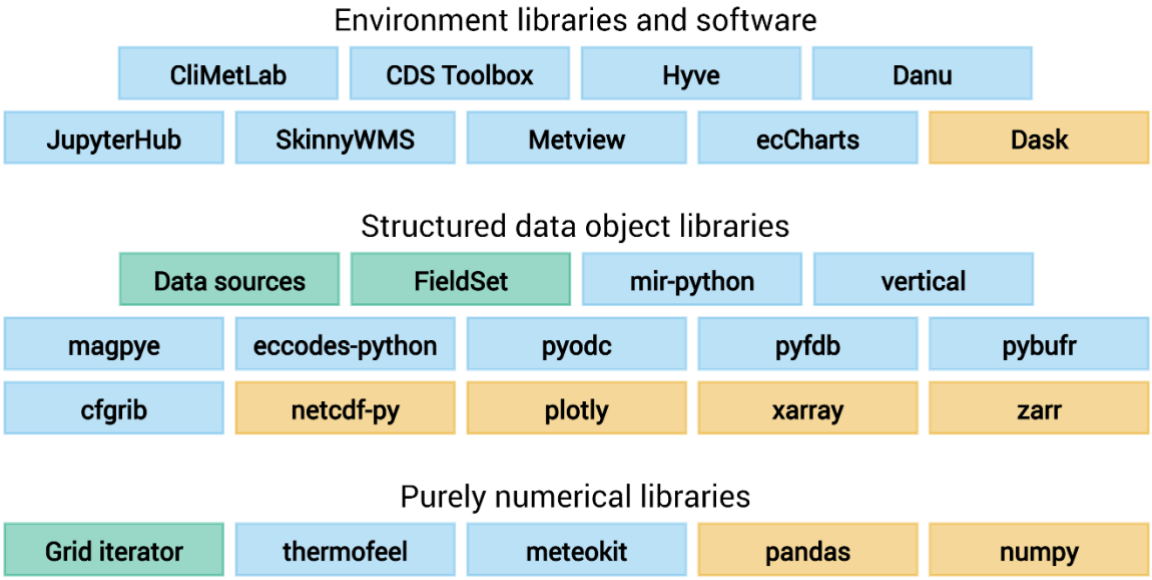


Figure 3: The Python software stack being refactored in the SPICE project This is a work in progress and far from complete, it will be subject to alterations in the years to come. Components in Green are new. Components in Blue exist and are developed by ECMWF and will be part of the refactoring effortsfc. Components in Orange are open-source and provided by the community.

Action Plan

The main guiding principle of the action plan is to develop more "Pythonic" interfaces to existing components, allowing seamless integration with existing community software. These are typically wrappers around the low-level compiled libraries (e.g., *pybufr*, *pyfdb* or *mir-python*).

We plan to incorporate the usage of well-supported software packages in our software stack, that can serve as building blocks to other higher-level meteorological specific functions. These will, together with other packages, serve as a supporting layer of pure numerical libraries, which implement algorithms on simple structures (e.g., *Numpy* arrays or *Pandas* frames). In Figure 3 this is depicted by the lower layer. These packages will be data format independent, hopefully maximising their reuse in different execution contexts. The middle layer will provide the structured data handling functionality, allowing integration with different data formats, and leveraging multiple community packages. In both these layers, we plan to develop a few new components, which applications and services can rely on to achieve certain functionalities:

Grid Iterator (working title), a new low-level, high efficiency compiled library, that will serve as the definition of the grids used in the Earth System Models. This will serve to consolidate code that over the years is repeated in a multitude of packages, with some resulting inconsistencies.

Data Sources (working title), a new set of libraries abstracting the access to data, independently of where the desired source is. This aims to serve both HPC batch-type workloads which access fast storage such as the FDB as well as interactive users that may run on Cloud systems, e.g., an online Jupyter notebook.

FieldSet, is a lift-off of the functionality that currently exists inside Metview to handle sets of Fields, manipulating them using scientific metadata and providing the abstraction level that is familiar to users of ECMWF data. Used in conjunction with Data Sources, many technical details of data

location and file format will be handled transparently. The Fieldset is intended to integrate well with Python community software.

In the top layer of Figure 3, we can see the application and services that will use the underlying plugins, composing them in unique ways to deliver services -- from hydrological post-processing (*Danu*); to machine learning data management (*CliMetLab*); passing through reliable hallmarks of ECMWF software such as Metview and ecCharts.

Milestones

2023: First version of Grid Iterator, Data Sources and FieldSet libraries
2024: All Python software packages open sourced and integrated with community development practices (open development principles)
2025: Generalised usage of the new concepts like Data Sources and FieldSet, and adopted into systems like Metview and Danu

Interactions

As a horizontal area of intervention, this work supports many other projects and cross cuts all development activities. It is deeply related to the refactoring of the *web framework*, the development of the *post-processing* project and the developments coming to the *Metview environment*, described later.

Once the components have a minimal usable functionality, we plan to open source them for interaction with the community, hoping this will eventually constitute a series of components that can be reused in contexts other than ECMWF services.

2.3. Data Encoding and Decoding Libraries

Providing forecast data is at the core of ECMWF's mission. It is crucial that ECMWF supplies ways to encode and decode the data in formats that are appropriate to the specificities of that data, such as WMO mandated formats. In this section we address the strategy for the low-level software that provides the encoding and decoding of all mission critical ECMWF data. It is separated into two major efforts aligned with two separate software packages: the first handles the coding of gridded forecast data and observations in BUFR format, whilst the second handles the coding of observation data, in the form that the IFS model handles (ODB).

Handling WMO GRIB/BUFR with ecCodes

Status

The software package ecCodes handles GRIB and BUFR data in such a way that provides a consistent interface no matter the edition of GRIB or BUFR used. It is written in the C language and provides a Python interface that maps directly to its C interface. For BUFR decoding we additionally have the legacy BUFRDC package, which undergoes only essential maintenance.

Motivation

As a library written in the C language, ecCodes has required much code of its own to implement basic data structures and operations. It has also been unable to benefit from developments in ECMWF's C++ packages, as their code could not be re-used inside it. This situation has led to duplication of code between packages and accrual of technical debt. Additionally, its Python interface is very tightly tied to the C interface and does not take advantage of the language elements that Python brings.

Resource pressures will push an emphasis on deprecating facilities that are hard to maintain. Python 2 has not been updated in two years, and niche architectures such as 32-bit platforms are hard to justify in a cost-benefit analysis. Reducing our support for such platforms will reduce technical debt and allow development time to be better focused on the main platforms that benefit most users. Support for ecCodes on the native Windows platform generates additional burden and will need to be carefully considered and properly defined.

The legacy BUFRDC package puts an additional burden in maintenance, and we do not have the resources to fix any major issues with it, thus leaving users vulnerable to any deficiencies in the package. Efforts must be made to remove any remaining obstacles that prevent the last users from migrating to ecCodes for their BUFR handling.

Action Plan

We see the future of ecCodes as still providing the existing C interface, but internally implemented as C++. This will allow for the removal of much code in favour of functionality supplied and optimised by the C++ STL (Standard Template Library). It will also allow for greater sharing of code and would open the possibility of being more involved in the software refactoring project (SPICE), using the proposed Grid Iterator library instead of its own code. The first step, which is to test the feasibility of building with a C++ compiler, has already been done, with favourable results.

Support for Python 2 and 32-bit architectures such as i686 will be dropped, enabling legacy code to be removed and reducing the time spent on hard-to-support platforms. Support for native Windows platforms will be defined after a requirement gathering exercise in consultation with users.

We will work with users of BUFRDC to ensure that the features and performance of ecCodes for BUFR handling are sufficient to enable migration. We will also give a reasonable timeframe before dropping all support for BUFRDC, to allow ECMWF's operational jobs to either be migrated, or expire in cases of legacy data sources.

The Python interface to ecCodes will gain a new, higher-level interface to make dealing with GRIB and BUFR messages from disk and memory simpler for users.

Milestones

2023: Release of new high-level Python interface for ecCodes
2023: Start building ecCodes as a C++ package and drop support for selected platforms
2024: Replace ecCodes' own iterator code with calls to the new Grid Iterator library from the SPICE project
2025: End all support for BUFRDC

Interactions

As ecCodes is such a fundamental component of our software stack, changes will be undertaken with much interaction across the ECMWF organisation. Liaison with Member and Co-operating State users and representatives will also be undertaken. ecCodes will also be fundamental in the project to move all ECMWF's data to GRIB2, involving much interaction with the ECMWF Data Governance process and related to WMO standardisation activities.

Handling of ODB observation data with ODC

Status

ECMWF has replaced the old ODB-API software with a new package for ODB-2 support, named *odc*. This came with several improvements, notably:

- Support for strings longer than 8 characters, enabling support for WIGOS identifiers
- New interfaces in C and Fortran, with consistent error handling
- Decoding data into, and encoding data from custom memory layouts

ODB-API has been deprecated and is now unsupported. A subset of the old ODB-API interfaces has been retained for a transitional period.

Alongside odc, ECMWF have released a Python library for working with ODB-2 data, and which integrates elegantly with *pandas* and *numpy*. This comes in two flavours; a Python-only library, pyodc, and a wrapper library, codc, which provides exactly the same interface, but offloads work to the compiled odc library for improved performance. These libraries present a relatively minimalist API, supporting only encoding and decoding data, and interrogation of the structure of ODB-2 data streams.

Motivation

We would like to encourage the use of the Python interfaces for researchers investigating and experimenting with observation data sets. Feedback from users has indicated that the SQL-like filtering functionality available in the odc command line tools, and through the MARS interface, is extremely useful, and its absence is a blocker to people adopting the Python interfaces. Although the intent was to build a thin encoder/decoder, the absence of this filtering functionality is a blocker to further uptake of the Python interface.

We would like to make use of pyodc and codc to support ongoing work with observations with external partners. This will assist them in developing clean and modern software tools and facilitate the use of novel observations.

Action Plan

We will implement the SQL filtering functionality in the codc module, alongside any necessary supporting code in the ODC C API. We note that this will result in a divergence between the functionality presented by pyodc and codc, as it will not be feasible to implement the SQL filtering in the pure Python module. The interface will be kept the same but users of pyodc will be encouraged to interoperate with Numpy and Pandas community software for filtering.

Once the use of data from the Observation Store is stabilised in the operational pipeline, we will work to remove the old-style deprecated interfaces, retained from ODB-API.

In addition, we will build a Python-based infrastructure for ingestion, quality control and encoding of high-frequency unconventional observations.

Milestones

- **2023:** Support implementation of the first version of the Observation Store
- **2023:** SQL-like filtering available through odc API
- **2025:** Infrastructure for ingestion, filtering, quality control and encoding of high-frequency unconventional observations

- **2026**: Removal of deprecated old-style ODB-API interfaces

Interactions

Whilst the Observation Store software uses odc, it is primarily built on the FDB software stack. Configuring this software will involve strong interactions with the ECMWF Data Governance process. Integrating the Observation Store into the forecast production pipeline, for direct ingestion of observations into the model will involves strong interactions with the COPE project.

The iChange and TRIGGER European projects will involve the acquisition, data governance, encoding, filtering and quality control of novel unconventional observations. The Destination Earth programme also contains work to bring higher resolution data into the model, which will have implications for observation encoding.

2.4. Visualisation Software

Status

Magics, ECMWF's meteorologically oriented visualisation software, has been developed and maintained for the last 40 years. It provides an easy and fast way to visualise data in GRIB, BUFR and NetCDF format. It is the graphical kernel of Metview and ecCharts, and is heavily used at ECMWF and some Member and Co-operating States. Its API provides users with a long list of parameters that give them the ability to tailor their plot but can be perceived as difficult to learn for a new generation of Python scientists used to working with packages such as matplotlib. It will require further maintenance and development to tackle the challenge of the data resolution increase.

Motivation

We strongly believe that we still need to provide our users with a meteorologically oriented package that will hide some of the complexities of handling meteorological data, but it is time to review what we offer, to see if it is still fit for purpose and ready for the next challenges -- and to imagine the visualisation component of the SPICE project.

Magpye and *bluejay* are a first attempt to create these lightweight Python modules, easily installed from Pypi, from day 1 with open documentation, and a gallery of Jupyter notebooks. Their final goal is to provide meteorologically oriented functions to help users to get a quick view of their data. Both packages have a small learning curve for newcomers due their Pythonic/matplotlib approach and will be able to display any type of data, provided the data implements some functionality needed for the specific visualisation. These mechanisms will ensure a good and light-weight interoperability with other SPICE components.

In a first stage, we envisage to delegate to plotly and matplotlib all the graphs functionalities (bluejay), and to Magics all the geographical visualisations (magpye), taking advantage of all the efforts that have been put in over the years in Magics to detect the most appropriate style to display the data. This approach will give us the freedom to review this decision and use new packages if they prove to be more suitable in the future and ensure a smooth transition.

Action Plan

The first action is to review the requirements for a graphical software such as Magics and be ready to rationalise the requirements that were accrued over the years. We will then need to objectively review the advantages/disadvantages of further developing and maintaining Magics, by investigating the possibility of using MIR to speed up the visualisation of high-resolution data, and the use of graphical Python packages.

In the meantime, a full analysis of magpye/bluejay should be done, to make sure that the offered solution will not only work for the next generation of the CADS Toolbox, but also for other ECMWF users, while ensuring their interoperability with other SPICE components. In tandem, we will also refactor applications such as SkinnyWMS or CliMetlab to make use of them, to minimise migration and maintenance.

Milestones

2023: Design the high-level interfaces of magpye and bluejay
2023: Evaluate the implications of moving away from Magics, and Plan the transition of Magics
2024: Begin the transition plan for Magics

Interactions

Magics is used by a large number of users and in operational systems such as Metview and ecCharts, Any change should be considered carefully: interaction with users and developers is key. To succeed, any decision in its future will be taken in close cooperation and agreement with the Metview and ecCharts developers and the transition carefully planned. This transition phase will have some challenging but positive aspects, as some historical plots will need a complete rewrite, reducing our technical debt.

Magpye and bluejay will be two important components of SPICE and should be carefully designed with users in mind as well as easy maintenance. We should be ready to interact and contribute with the matplotlib community.

2.5. IO-server and On-the-fly Processing Pipelines

Status

Currently, the IFS features an IO-server that was implemented by Météo-France and further extended by ECMWF. This IO-server is written in Fortran and required significant work to support the Wave model output. The EU funded NEXTGenIO project provided the opportunity to do the Wave model migration, and the extension of the IO-server for ensemble forecasts. This was critical for the scalability of the ensemble forecasts in CY45R1. Whilst the current IFS IO-server performs very well, it is quite inflexible and was laborious to extend.

The ocean model currently used by ECMWF (NEMO v3), uses another IO-server technology (XIOS) which has features for on-the-fly processing which are used for computing temporal statistics and cumulative parameters. However, XIOS only supports NetCDF and a strictly file-based output. A new solution is required to support a message-oriented protocol and the GRIB format to allow the proper storage of ocean fields in FDB and MARS, and enable synergies with the remaining ECMWF workflow, and achieve scalability at very high resolutions.

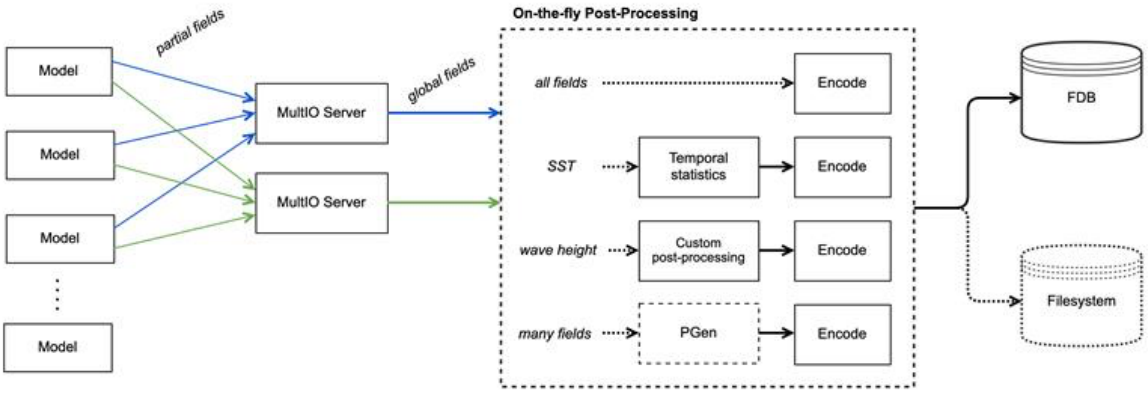


Figure 4: Design of the new MultIO server showing various on-the-fly post-processing pipelines for different fields, with flexible output to different types of data storage.

Motivation

ECMWF is extending the existing MultIO software with IO-server functionality, heavily inspired by the efficient aggregation mechanisms of the current IFS IO-server and the flexible post-processing pipelines available in XIOS (see Figure 4). The new MultIO IO-server handles aggregation of fields from distributed compute nodes, using flexible transport protocols (e.g., MPI, TCP) and allow direct writing to different types of data storage (including FDB/MARS). Furthermore, the MultIO IO-server supports user programmable pipelines for on-the-fly post-processing, allowing for generation of additional fields and products, such as temporal statistics, without having to write-and-read-back from the IO system. This is crucial technology to enable the handling of the upcoming very high-resolution data sets.

Action Plan

The MultIO IO-server will initially be used by the NEMO Ocean model from v4 onwards, currently scheduled for CY49. The new IO-server will also be used for the new FVM dynamic core model being developed at ECMWF. After demonstrating its performance and flexibility, the MultIO IO-server will be

adopted for use by the atmosphere and wave models' output. This is expected to be a contribution from the DestinE project.

The programmable pipelines will allow different post-processing to be applied to different types of fields, and we intend to execute much more of our post-processing stack on-the-fly using this mechanism. Infero, a new software package developed for unified execution of machine-learned inference models, currently runs within IFS and executes on in-memory data from IFS. We expect to run Infero within the programmable MultIO post-processing pipelines, therefore executing ML models on global fields very effectively without touching the I/O system.

Furthermore, ECMWF will evaluate the feasibility of executing part of the product generation (PGen) workload inside the MultIO processing pipelines, seeking to massively improve I/O efficiency and potentially allow earlier creation of products. This was already partially demonstrated within the EU funded MAESTRO project, with very promising results. Further work is needed to operationalise this prototype and demonstrate it at scale.

The MultIO model will also be developed to support multi-threaded asynchronous encoding of data, to further minimise the impact of output on the model runtime. This will support better configuration of HPC job geometries to maximise use of available physical cores and hyper-threading on modern systems.

Milestones

2023: MultIO used for NEMO v4 output in CY49
2025: MultIO used for IFS atmospheric and wave model output
2027: Parts of PGen executed on-the-fly as part of MultIO programmable pipelines

Interactions

MultIO server activity has strong interactions with IFS and NEMO developments, in particular, MultIO interacts with the Atlas library. The post-processing pipelines will impact the PGen software. There are also opportunities to interact with the machine learning efforts at ECMWF and data compression developments. Through Destination Earth there will be collaboration with other partners and external communities, since MultIO and its on-the-fly post-processing features are a core part of the Digital Twin Engine. MultIO is open source, and collaboration will be encouraged with all interested parties, including Member and Co-Operating States.

2.6. Data Dissemination and Acquisition

Status

Data dissemination and acquisition at ECMWF is handled by the ECMWF Production Data Store (ECPDS). This is now a very mature software, which has been in operational use for several years. However, the operational context is shifting. As a result of the ECPDS extension to the Scalable Acquisition and Pre-Processing (SAPP) system Optional Programme, various Member States and collaborations intend to use ECPDS for their data acquisition, dissemination, and inter-site data movements. Work to support the needs of the Member States in this Optional Programme will drive additional upcoming work.

Motivation

As a component that interacts with external data sources and users, ECPDS has a relatively high level of business-as-usual development. There is a constant stream of work supporting new network transport

protocols and data endpoint types. This is especially true given the offerings of current cloud providers, whose interfaces shift regularly.

The SAPP Optional Programme requires the development and deployment of a standalone ECPDS distribution – that is, one that does not require ECMWF's operational infrastructure to work. The decoupling of ECPDS from ECMWF's infrastructure, and its adaption to the needs of the Member States will drive most of the immediately upcoming work.

The long-term goal of ECPDS has always been to support data transfers in a scalable manner, to a range of locations. As our data volumes continue to increase, alongside the range of destinations, we need to improve our data transfer capabilities and performance.

Action Plan

Development work is required to support integration of ECPDS into Member States existing IT infrastructure. This particularly includes monitoring, authentication and accounting functionality. Further, as ECPDS has been developed and deployed by a tight-knit development and operations team, the external facing documentation is insufficient to support external operational teams and will need significant work. In conjunction with that, ECPDS will be open sourced to make external access and deployment more straightforward.

Member States in the SAPP Optional Programme wish to use ECPDS to support synchronisation between multiple sites. This is not currently supported by ECPDS, but is a natural fit to its capabilities, and will also be a potentially useful functionality to support the Destination Earth programme by linking ECMWF's data centre and the EuroHPC data centres. This functionality will need to be developed once the initial integration of the standalone ECPDS with Member States infrastructures is completed.

The WMO is working towards the replacement of the Global Telecommunication System (GTS) with the WMO Information System (WIS) 2.0. This will present a publish-subscribe (pub/sub) interface for the acquisition and distribution of observations. ECMWF already has support for Malawi prototype, operated by the WMO, which is making available observations from a selection of African countries. But this support needs to be extended to support wider uptake of this technology.

Milestones

- **2023:** User and administrator documentation of ECPDS for Member States
- **2024:** ECPDS fully integrated into at least one Member State's datacentre
- **2024:** ECPDS open-sourced
- **2024:** Full-featured use of WIS 2.0
- **2025:** Inter data-centre synchronisation supported

Interactions

ECPDS development has several ongoing business-as-usual interactions with other teams. Most notably, there is a strong link to operations run by the Forecast Delivery Team, but also with the operational teams involved with acquisition and product generation. The Copernicus Atmosphere Monitoring Service (CAMS) also makes use of ECPDS for observation retrieval and distributing data to their users through the data portal.

In terms of external development, ECPDS development has strong links to the Member States involved in the SAPP Optional Programme, especially Ireland and the members of the UWC West consortium.

ECPDS is likely to be used for acquisition of novel, unconventional observations in the iChange and TRIGGER projects. The Destination Earth project involves attention to novel satellite observations and may use ECPDS to synchronise data between ECMWF's data centre and EuroHPC computing facilities.

ECPDS will be heavily impacted by, and involved in, the shift towards provision of open data.

2.7. Observational Data Pre-Processing

Status

ECMWF's Scalable Acquisition and Pre-Processing system (SAPP), is a critical and essential component of the NWP processing chain, delivering timely observational BUFR data to the IFS operational assimilation.

The majority of business-as-usual activity is currently focused on introducing new observations in operations; this includes configuring data acquisition (via ECPDS service) and developing decoding and processing workflows orchestrated by SAPP.

The SAPP Optional Programme (SAPP OP), launched in 2019, allowed several participating NMSs to start running customized SAPP virtual instances in their own local infrastructure with good results being reported both in terms of system availability and increased number of assimilated observations and, therefore, improved forecast quality.

Motivation

Over the last few years, new technological and operational requirements have emerged from BOND migration and SAPP Optional Programme. The following key areas of maintenance and evolution have been identified:

Address technical debt by decommissioning or migrating legacy code and workflows and continue refactoring and decoupling software from ECMWF infrastructure, allowing for higher local customization (relevant to SAPP OP).
Extend and improve system provisioning to virtual/cloud-based platforms, to enable and facilitate scalability, automated testing and Continuous Development and integration.
Support WMO driven developments (migration to WIGOS identifier, WIS 2.0).
Support IFS data processing developments (e.g., COPE) and extend range of data formats and observations handled, to include novel and higher spatial/temporal resolution data (DestinE).

Action Plan

Several migration and porting activities are planned or already under way to address and reduce technical debt: Python2 to Python3, and PGI to GNU Fortran code migrations will be completed ahead of operational implementation in Bologna.

The replacement of Legacy BUFRDC Fortran software with equivalent ecCodes Fortran/Python programs is also under way and internal Python modules used for BUFR processing are being refactored to simplify decoders customization and introduce higher level of abstractions for decoding/encoding and filtering steps.

In terms of system provisioning, a SAPP docker system (developed within the scope of SAPP OP) is being consolidated and will be finalized after the BOND migration. The containerized solution, together with the automation of virtual instance provisioning, is expected to vastly simplify deployment and integration of the system both on ECMWF and Member States infrastructures.

As WMO Information System (WIS) 2.0 will progressively replace Global Telecommunication System (GTS) some work will also be required to adapt SAPP acquisition stage to use WIS catalogue and topics metadata in place of GTS bulletin headers for decoders routing logic. Monitoring tools will also be enhanced to improve tracking of WIGOS Station identifier uptake for different observation types and reporting of BUFR templates usage in operationally exchanged data.

SAPP extraction stage, feeding IFS Data Assimilation, will also be reviewed to meet the needs of Continuous Observation Processing Environment (COPE) project; the current plan is to enable hourly/sub-hourly BUFR extractions in IFS cycle 49r1.

Finally, incoming data format handling will also be extended to accommodate more JSON/CSV/NetCDF to BUFR workflows in support of High-Res Data and novel/unconventional observation processing as requested in the frame of different evolution projects (e.g., SAPP OP, DestinE).

Milestones

- **2023:** SAPP docker system available; provisioning on cloud/virtual infrastructures
- **2024:** COPE extractions in operations
- **2024:** WIS 2.0 data acquisition (thru ECPDS-ACQ)
- **2025:** Finalize SAPP decoders BUFRDC migration

Interactions

SAPP strongly relies on ecCodes for its BUFR decoding/encoding layer and gets all incoming observational data from the ECPDS system; as such strong collaboration is in place to follow and adapt to new data format and acquisition developments (EUMETcast, WIS 2.0, Observational data governance). Similarly, interactions with WMO for data governance activities and with internal teams and systems involved in observation verification and assimilation can drive the implementation of new processing and monitoring features (COPE project, WIGOS identifier handling and monitoring).

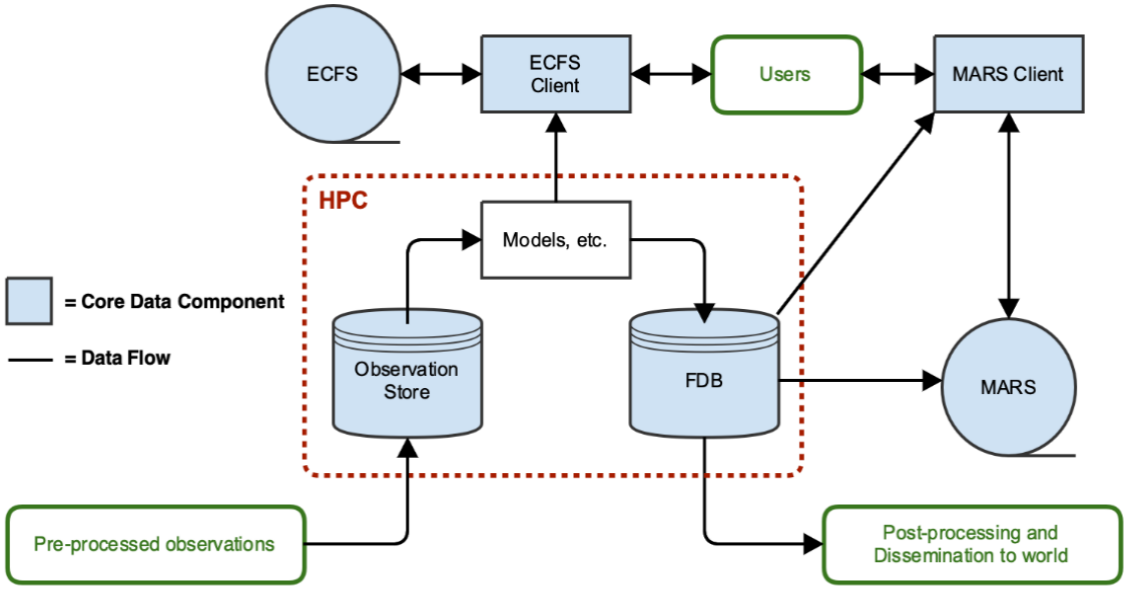
2.8. Core Data Storage Software

Status

One of ECMWF's great strengths is its metadata-driven workflows. Data is stored in self-describing data formats and can be handled according to globally-unique identifying metadata through all parts of our workflows. For examples, not only is storage and retrieval of meteorological data metadata driven, but also post-processing and many categories of custom computation and dissemination of products to users.

A large part of this functionality is driven by the MARS ecosystem, comprising the MARS server (holding ECMWF's long-term meteorological archive), the FDB (an in-HPC, high-performance object store) and the MARS client. Additionally, ECFS provides an unstructured archive for data which is not self-describing meteorological data, or otherwise sits outside of the metadata-driven workflows.

Version 5 of the FDB was deployed into operations at ECMWF in 2019, after a long period of development.



This brought substantial convergence between the FDB and the rest of the MARS ecosystem, as well as performance and semantic improvements on the parallel filesystem. But the FDB has also brought a great deal of forward-looking flexibility to be configured in different ways and to support different backend technologies.

Recent developments have also focussed on the convergence of the MARS and ECFS software and operational environments. In 2019 there was no commonality at all between these two services, which now share deployment and administrative environments and large chunks of their codebases.

Motivation

Data handling underpins ECMWF's operational systems, giving us leading-edge performance and scalability on the HPC (through the FDB), consistent and semantic access to data throughout our workflows and integration of our long-term meteorological archive (MARS) into the same data ecosystem as current data. But the data and storage landscapes are constantly changing, and we need to address several driving factors to support ECMWF's strategy:

1. Resolution and data diversity increases drive constant increases in data volume.
2. The storage landscape is becoming more heterogeneous, and we wish to be able to make use of new technologies on the market.
3. HPC and Cloud systems are gradually converging, and we wish to enable and support use of our data from the cloud.
4. Open data is increasingly important, and comes with higher volumes of data access, and different access patterns and semantics.

Beyond even these driving factors, we wish to increase the flexibility of the configurations that can be supported by the FDB, which will facilitate the development of new backends to the FDB software. We also

wish to support a wider range of data access paradigms, particularly the extraction of data across different axes and subsetting patterns to those currently supported, and to support the storage of novel types of data using the same access technology.

Finally, having seen the benefits of the work done so far, we wish to continue and extend the convergence between the MARS and ECFS domains, and to continue to excise and replace legacy software components with more modern components. This will reduce the long-term maintenance and development effort required, and also reduce the overhead on the operational teams who administer and run these services 24/7.

Action Plan

Larger data volumes will require the use of novel storage technology that is coming onto the market, including solid state storage and storage class memory, and high-performance object storage technology built on them, as well as cloud storage paradigms. We will expend significant effort building backends to support, and evaluate, a range of hardware backends. This will give us significant flexibility during procurements going forwards. We will also restructure the indexing and configuration of the FDB to allow the use of multiple data backends simultaneously, and to support migration of data between backends.

As data volumes increase, the impact of using HPC and other resources to move data between data system components becomes more significant. We will bring in usage of direct transfers of data from the FDB to MARS, and between storage tiers and systems within the FDB. We will also implement a Volume FDB, to store large volumes of research data on non-tape storage outside the HPC, and avoid unnecessary transfer and archival of research data with a short lifespan.

At present, ECMWF's workflows are largely driven by single users, and the permissions and access model is delegated to the POSIX filesystem. To support non-POSIX backends, and use of the FDB in cloud and more diverse user environments the FDB will need to support explicit user authentication and authorisation. This will likely involve reworking the remote-FDB protocol and may involve a thin FDB microservice layer for indexing and for data storage.

The massively increased data volumes implied by increases in model resolution will drive a need to serve more data at reduced resolution. The open data strategy combined with access from the European Weather Cloud drive very different access patterns, slicing data across different axes. We will extend the FDB ecosystem to be able to store reduced resolution or re-gridded copies of data, in secondary locations, to support resource-efficient access. We will also build support for directly accessing slices and geometrically defined subsets of the data, building on developments made in EU research projects (Polytope).

We aim to take the opportunities presented by required performance improvements and new functionality to reduce duplication in our software stack, whilst removing legacy code. MARS, FDB and ECFS began as entirely separate codebases, and are undergoing a continuous convergence. We expect to standardise the MARS language handling across our stack and to replace much of the server machinery in ECFS with that from MARS. The new Observation Store (see Observation Handling above) is built using the FDB technology.

The MARS and ECFS clients are now old pieces of software. We will reduce our long-term maintenance overhead by (re)implementing MARS and ECFS clients using modern languages and paradigms. These will provide a programmatic, python-based interface to improve their integration into our workflows. By reusing tested software components between the FDB, MARS server and the new clients this will improve robustness with lower development overhead. Replacing the client-server communication protocol, currently

based on FTP with that used in MARS will also enable significant performance and semantic improvements in ECFS when handling large data transfers.

Milestones

2023: Implement Volume FDB, and Observation Store for use in research mode
2023: Develop DAOS and MOTR backends for the FDB
2024: Multiple backends per database and explicit user and permissions management in the FDB
2024: First versions of the new MARS and ECFS Clients
2025: Feature extraction support in the FDB
2025: Direct transfers (FDB to FDB, MARS to MARS and FDB to MARS)

Interactions

The Core Data Storage Software forms a pipeline that links many groups and activities at ECMWF. Development work on these software packages involves collaborating with and balancing the needs of these different groups.

Developments to our Core Data Storage Software underpins many of the goals and developments in ECMWF's externally facing projects. The ACROSS and IO-SEA projects involve direct work on the FDB to support novel storage backends and data flows. The Destination Earth project will involve significant development work on the FDB software, and deployment of the FDB and Observation Store into novel contexts, storing novel data. We also anticipate collaboration with German Climate Computing Center (DKRZ), MeteoSuisse and CSCS, amongst others, to support the FDB and metadata-driven workflows.

2.9. Post-Processing Framework

Status

Multiple activities within ECMWF depend on time-critical post-processing suites which run downstream of IFS. These suites are usually not compatible with PGen (Product Generation software), because instead of 1-field-to-many-products, these suites perform statistics or climatology across many fields (e.g., clustering products, shift-of-tails, extreme forecast index). These suites have been developed disparately across the centre, and many suffer from scalability issues which will prevent scaling to the planned ensemble forecast at 9 km resolution.

Motivation

We aim to develop a framework for post-processing suites, which will consolidate post-processing functionality under a single environment, allying modern and scalable computer science technologies with best scientific practices. This will result in higher performance, greater scalability and improved I/O efficiency. Once this framework is in place, we expect that the maintenance and portability of this framework to be much increased. For example, it will be possible to reuse the same consolidated procedure between post-processing the operational model output and that of a research experiment (which is only achieved today with laborious ad-hoc procedures).

Action Plan

The first goal is to port the various tools used in these suites to leverage the refactored software stack introduced earlier, the harmonised Python ecosystem developed by the SPICE project. This will improve the robustness of the tools, ensuring the same methods and algorithms are used throughout, and improve their maintainability. In many cases, this will already improve the performance and scalability of these tools.

The second goal is to harmonise the execution of the suites, by refactoring them under a common Pyflow framework. This will improve the operational management of these suites, whilst also enabling simple execution within research experiments (including via IFSHub). It will also bring parallelism to essential suites, such that they are ready to scale to 9-kilometre ensembles by CY48R1.

With this post-processing framework in place, we will be in a strong position to explore much more efficient execution of these suites, whether through optimizing I/O or by utilising distributed parallel computing frameworks. This will enable scaling of these suites beyond PiB-scale forecasts.

Milestones

2023: Port time-critical, unscalable post-processing suites for 9-kilometre ensemble using the post-processing framework, demonstrating its effectiveness
2025: Pioneer and standardise post-processing suite development across the centre
2025: Enhance the post-processing framework with a robust and scalable parallel execution framework

Interactions

There will be strong collaboration across the ECMWF as multiple teams contribute with scientific functionality into the framework. The suite refactoring will rely on, and help to develop, the Python components of the SPICE project. The work will also complement IFSHub, allowing execution of suites within RD experiments.

2.10. Metview Environment

Status

Metview is ECMWF's user-facing software package that provides a single environment for accessing, processing and visualising data. It includes a graphical user interface (GUI), a Python interface and its own Macro programming language. Over its 30 years of development, Metview has accumulated much code for processing model and observation data. While its original design made good use of the libraries available at the time (e.g. the MARS client with its fieldset handling and service-oriented architecture), many new libraries have since evolved, sometimes needing to replicate some of Metview's functionality while remaining independent of it. This duplication of code leads to technical debt, and different implementations can lead to software packages producing slightly different results for the same computations. Most of the current software stack that Metview relies on is hard-coded to use double-precision floating point arrays, making manipulation of high-resolution data (e.g. 1km) very expensive in terms of memory usage. Metview's visualisation capabilities come from the Magics library, with a close coupling of code.

Motivation

As we refactor the software stack there is a chance to rationalise the situation with Metview, with code and algorithms from Metview being contributed to new software components that Metview and other packages can use. For example, many thermodynamic functions are already being contributed to meteokit, and once the software refactoring project is underway, Metview's own code for these computations can be removed, and Metview (and other packages) can instead rely on the functions donated to meteokit. The same is true for other components in the upcoming software stack. Metview will continue to provide users with a high-level way of interacting with our data, including its GUI and improved ways of working in Jupyter notebooks.

With an upcoming new MARS client, much of Metview’s code must be refactored to adopt it while retaining its current functionality. This will be an opportunity to reduce the amount of code in Metview and increase its consistency.

As data volumes increase, so too must Metview's capacity to handle them. We are already seeing 1km grids from Research, and even the resolution increase of the ENS forecast could have an impact on data handling. Metview must be able to continue processing and visualising such fields efficiently with existing hardware.

Metview’s close coupling with Magics currently imposes some constraints on how we package both software packages, leading to extra maintenance work. Having a looser coupling will reduce overall maintenance and allow both packages to evolve more independently.

As Metview’s Python interface becomes more widely used due to its greater expressiveness and flexibility, the original Macro language will become legacy, and supporting both will become a duplication of effort. The Python ecosystem will be the way forward, especially in the context of the functionality that the SPICE project components will be able to offer. We will first freeze development of the Macro language, following that with a decommissioning in due time.

A relatively new environment where Metview is being increasingly used is the Jupyter notebook. In this environment, Metview’s GUI is usually not available, creating a need for new components that allow data inspection inside the notebook.

Action Plan

We will use the opportunities provided by the SPICE project to (i) donate code and algorithms from Metview to new software components that can be used elsewhere, and (ii) re-base Metview on this more modern software stack.

Another refactoring will involve replacing the existing C-based MARS client with the new C++ based client as Metview's C++ interface to the MARS archive and to basic GRIB-based computations. This will bring Metview's binary components up to date as part of a well-tested software stack.

Further work will be done to enhance the usability of Metview inside Jupyter notebook environments, in particular interactive data inspection and plotting widgets.

The Magics plotting library contains a substantial amount of code that is used only by Metview's interactive plotting window. By extracting this code from Magics and making it part of Metview, maintenance and packaging of Magics will be eased, and Metview will gain more flexibility in implementation.

A phased approach over the coming years will deprecate and eventually remove the Macro language from Metview, starting with a freeze on new features. We will explore options that could assist users to migrate their scripts to Python.

We will ensure that Metview and all its components can handle the large volumes of data envisioned in future model upgrades by identifying and tackling bottlenecks. One component of this is to ensure that all packages can work with single precision floating point arrays if requested to do so, thus halving their memory usage.

Milestones

- **2023**: Freeze development of Metview's Macro language
- **2024**: Refactor of Metview to use new MARS client code
- **2024**: Enhanced Metview widgets for interoperability with notebook environments
- **2024**: Development of SPICE components required by Metview
- **2025**: Refactor Metview to use new SPICE components and reduce strong dependency on Magics
- **2027**: Removal of Macro language from Metview installations

Interactions

Important stakeholders in these plans are all users of Metview, which include Member and Co-operating State users. The work to reduce memory consumption will require interactions with the DestinE project.

2.11. Web Framework

Status

ECMWF and Copernicus are offering many web applications to their users such as Opencharts/ecCharts, WebMARS for ECMWF and EFAS/CDS/ADS for Copernicus. Currently, while most of them share the same general principles and technologies, they are still maintained and developed in isolation. This situation evolved slowly over recent years, as the web became a more natural way to interact and provide ECMWF services. It is now opportune to find synergies between applications and consolidate common functionalities in reusable components.

Motivation

We plan to capitalise on the experience the web team has built in developing these operational applications and start working on reusability of components, interoperability of services and shared technologies.

We will identify reusable components that will make us more efficient in tackling future challenges such as the increase of the ensemble resolution.

A strong focus will be on interoperability, as this will play an important role on projects such as IFSHub or CADS where, from a single friendly environment, users will be able to interact and connect several services they need for their daily work. An improved interoperability with external systems such as WEkEO will facilitate interactions and exchanges.

Finally, by sharing components built on the shared technologies, we will benefit from a consolidated infrastructure and software. This will facilitate the operational monitoring and resolution of issues by increasing visibility, and we will extend our knowledge base.

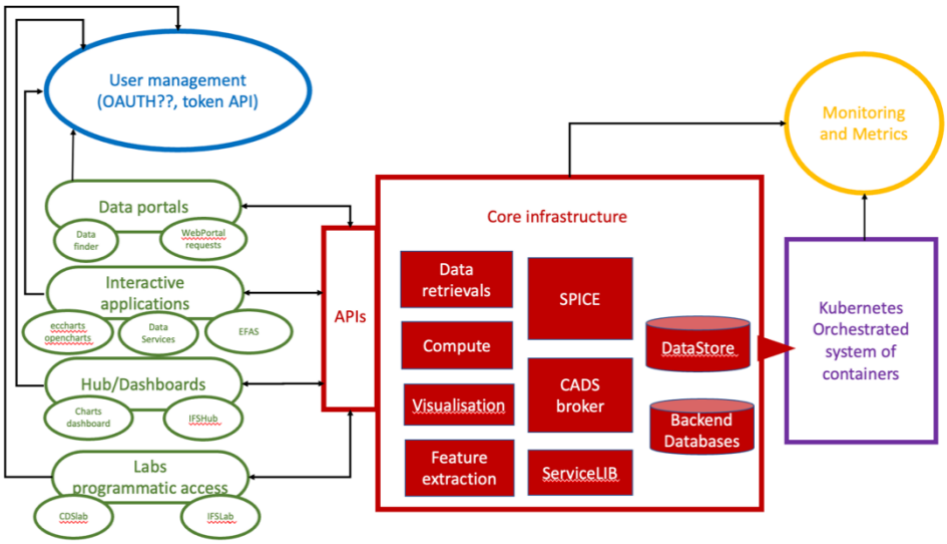


Figure 5: Schema of the web framework: 4 categories of web services using a large set of well documented APIs, themselves relying on a core infrastructure redesigned to use and contribute to SPICE

Action Plan

The first action is to review the web services (User facing, and internal ones) we are already offering, documenting clearly their API using the OpenAPI standard. We should make use of the existing standard tools available: they offer convenient ways to present the documentation and functionalities to the APIs via OpenAPI. Once done, we will be able to refactor them one by one as necessary, and make use of some of them to create more complex applications such as IFSHub.

During that phase we will also take the opportunity to review codes and Git repositories to identify reusable components that could be nicely packaged and become candidates for integration into the SPICE project.

This work, and the generalisation of the use of opensource solutions which are well documented and accepted by the broad community, will help us to reach a better interoperability between our diverse areas of activities.

Another important aspect to consider soon is the impact of ENS resolution upgrade on our on-demand services (ecCharts/opencharts) and in the long term the impact of a further increase in data resolution.

The web services have a lot of challenges; it is important to find a common approach to tackle complex issues such as cache, storage and authentication and improve our use of CI/CD by migrating to GitHub Enterprise and GitHub Actions.

We have new projects in the pipeline: improve the discoverability of ECMWF data, develop the next generation of the CADS Toolbox, create the next environment for researchers to achieve their daily work. It is important to take advantage of previous experiences and design if we want to offer our users a pleasant and consistent web experience.

Milestones

2023: Review web components, refactor WebMars and Data Layers
2023: Design IFSHub as a flexible platform to plugin components needed to researchers to perform their daily work.
2024: Launch of CADS (Copernicus founded)
2024: First version of IFSHub
2025: Operational IFSHub
2026: New ecCharts ready for high resolution data.

Interactions

The success for a web application is to work in close cooperation with its potential users to fulfil their expectations. The next 5 years will see the introduction of several brand-new systems: CADS, IFSHub and SPICE, that will require strong interactions and communications.

CADS as the new generation of the CDS will need to answer the requirements of a well-established community.

IFSHub will need a solid design, where the challenge is to create an intuitive platform for researchers, generic and flexible enough to allow easy integration of new components and interoperability between them.

3. Summary

This software strategy and subsequent implementation roadmap complement each other when seen holistically. The work in each Area of Intervention may stand on its own, but it creates synergies when all are brought to bear fruits. The area of *refactoring of software stack*, will provide new components and renewed flexibility in the software stack to support the other areas. The developments in *web framework* will provide improved flexibility to deliver user facing services with more reusable components. These services will be fed from products created with the new *post-processing* framework, which will draw from components refactored out of the *Metview environment*, and now available to run of alternative execution contexts like the CADS and the HPCF. The *Metview environment* will also profit from structural changes coming to the *visualisation software*, where we are certain to find multiple synergies with the new version of the CDS (CADS). The developments of *data dissemination and acquisition software* will support NMS using this system, but also look forward to tighter integration in the *core data storage software*, regarding projects in IoT observation data acquisition. Finally, the developments of the new *IO-server and on-the-fly processing pipelines* which are so close to the Earth System Model, are tightly linked to the remainder of the workflow in terms of data processing and encoding, to ensure that the overall workflow scales to forthcoming the data handling challenges.

This software strategy and roadmap will allow ECMWF to support our services more efficiently, whilst simultaneously preparing for the challenges that larger datasets will bring. It will also allow ECMWF to address the wider challenges of adapting its software infrastructure to the requirements deriving from the ECMWF Strategy 2021-2030.

3.1. Summary of Milestones

Since each area of intervention defined its own milestones, to provide an overall view of how these integrate, the milestones are thus collected here:

Year	Description	Area of Intervention
2023	First version of GridIterator, DataSources and FieldSet libraries	Refactoring of Software Stack
	MultIO used for NEMO v4 output in CY49	IO-server and On-the-fly Processing Pipelines
	Review web components, refactor WebMars and Data Layers	Web services
	Design IFSHub as a flexible platform to plugin components needed to researchers to perform their daily work	Web services
	Port time-critical, unscalable post-processing suites for 9-kilometre ensemble using the post-processing framework, demonstrating its effectiveness	Post-Processing
	Release of new high-level Python interface for ecCodes	Data Encoding and Decoding Libraries
	Start building ecCodes as a C++ package and drop support for selected platforms	Data Encoding and Decoding Libraries
	Support implementation of the first version of the Observation Store	Data Encoding and Decoding Libraries
	SQL-like filtering available through odc API	Data Encoding and Decoding Libraries
	Freeze development of Metview's Macro language	Metview Environment
	Design the high-level interfaces of magpye and bluejay	Visualisation Software
	Evaluate the implications of moving away from Magics and plan transition of Magics	Visualisation Software

	User and administrator documentation of ECPDS for member states	Data dissemination and acquisition
	Implement Volume FDB, and Observation Store for use in research mode	Core Data Storage Software
	Develop DAOS and MOTR backends for the FDB	Core Data Storage Software
2024	All Python software packages open sourced and integrated with community development practices	Refactoring of Software Stack
	Launch of CADS	Web services
	First version of IFSHub	Web services
	Replace ecCodes' own iterator code with calls to the new GridIterator library from the SPICE project	Data Encoding and Decoding Libraries
	Refactor of Metview to use new MARS client code	Metview Environment
	Enhanced Metview widgets for interoperability with notebook environments	Metview Environment
	Development of SPICE components required by Metview	Metview Environment
	Begin the transition plan for Magics	Visualisation Software
	ECPDS fully integrated into at least one member state's datacentre	Data dissemination and acquisition
	ECPDS open-sourced	Data dissemination and acquisition
	Full-featured use of WIS 2.0	Data dissemination and acquisition
	Multiple backends per database and explicit user and permissions management in the FDB	Core Data Storage Software
	First versions of the new MARS and ECFS Clients	Core Data Storage Software
2025	Generalised usage of the new concepts like DataSource and FieldSet, and adopted into systems like Metview and Danu	Refactoring of Software Stack

	MultIO used for IFS atmospheric and wave model output	IO-server and On-the-fly Processing Pipelines
	Operational IFSHub	Web services
	Pioneer and standardise post-processing suite development across the centre	Post-Processing
	Enhance the post-processing framework with a robust and scalable parallel execution framework	Post-Processing
	End all support for BUFRDC	Data Encoding and Decoding Libraries
	Infrastructure for ingestion, filtering, quality control and encoding of high-frequency unconventional observations	Data Encoding and Decoding Libraries
	Refactor Metview to use new SPICE components and reduce strong dependency on Magics	Metview Environment
	Inter data-centre synchronisation supported	Data dissemination and acquisition
	Feature extraction support in the FDB	Core Data Storage Software
	Direct transfers (FDB to FDB, MARS to MARS and FDB to MARS)	Core Data Storage Software
2026	New ecCharts ready for high resolution data	Web services
	Removal of deprecated old-style ODB-API interfaces	Data Encoding and Decoding Libraries
2027	Parts of PGen executed on-the-fly as part of MultIO programmable pipelines	IO-server and On-the-fly Processing Pipelines
	Removal of Macro language from Metview installations	Metview Environment

4. Glossary

Term	Description
bluejay	SPICE component, meteorological graph based on plotly
BUFRDC	ECMWF's legacy software for encoding/decoding BUFR data https://git.ecmwf.int/projects/MARS/repos/bufrdc_emoslib/browse
CADS	Next generation of the Copernicus Datastore https://cds.climate.copernicus.eu/#!/home https://ads.atmosphere.copernicus.eu/#!/home
CliMetLab	A Python package aiming at simplifying access to climate and meteorological datasets https://github.com/ecmwf/climetlab
COPE	Continuous Observation Processing Environment https://www.ecmwf.int/en/newsletter/158/meteorology/continuous-data-assimilation-ifs
Danu	ECMWF's python framework supporting the EFAS and GloFAS medium-range forecast product generation as well as flash flood models.
DestinE	Destination Earth Programme https://digital-strategy.ec.europa.eu/en/policies/destination-earth
DHS	ECMWF's Data Handling System https://www.ecmwf.int/en/computing/our-facilities/data-handling-system
ECFS	ECMWF's unstructured tape storage system https://git.ecmwf.int/projects/ECFS/repos/ecfs-server/browse https://git.ecmwf.int/projects/ECFS/repos/ecfs-client/browse
ecFlow	ECMWF's workflow manager https://github.com/ecmwf/ecflow
ecCharts	ECMWF's web service for weather forecast charts https://git.ecmwf.int/projects/ECCHARTS/repos/eccharts/browse
Code For Earth	ECMWF Summer of Weather Code https://codeforearth.ecmwf.int
EWC	European Weather Cloud https://www.europeanweather.cloud/

FDB	ECMWF's domain-specific object store for direct model output https://github.com/ecmwf/fdb
HPCF	ECMWF High Performance Computing Facility https://www.ecmwf.int/en/computing/our-facilities/supercomputer
IFSHub	A centralised web hub for interacting with research experiments at ECMWF https://www.ecmwf.int/en/newsletter/167/computing/ifshub-new-way-work-ifs-experiments
Infero	ECMWF's engine for running inference ML models in time-critical operations supporting multiple pluggable back-ends. https://github.com/ecmwf-projects/infero
Magics	ECMWF's visualisation library https://github.com/ecmwf/magics
magpye	SPICE component, based on Magics and Matplotlib https://github.com/ecmwf/magpye
matplotlib	Open-source visualisation package for Python https://github.com/matplotlib/matplotlib
MARS	ECMWF Meteorological Archival and Retrieval System https://git.ecmwf.int/projects/MARS/repos/mars-server/browse https://git.ecmwf.int/projects/MARS/repos/mars-client/browse
Metview	ECMWF's data analysis and visualisation environment https://git.ecmwf.int/projects/METV/repos/metview/browse
meteokit	Python library developed by ECMWF to consolidate functionalities common to multiple meteorological packages, aiming at performance and reusability. https://git.ecmwf.int/projects/ECSDK/repos/meteokit/browse
MultIO	ECMWF's next generation Multiplexing I/O asynchronous server https://github.com/ecmwf/multio
OpenAPI	Standard, language-agnostic interface to RESTful APIs https://www.openapis.org/
PGen	ECMWF's Product Generation software running in time-critical operations https://git.ecmwf.int/projects/PRODGEN/repos/pgen/browse

plotly	Open-source Graphing Library for Python https://github.com/plotly/plotly.py
Pyflow	Python based high-level description language for defining ecFlow https://git.ecmwf.int/projects/ECFLOW/repos/pyflow/browse
pypi	Repository of software for Python. https://pypi.org
readthedocs	Open-sourced free software documentation hosting platform https://readthedocs.org
SAPP	ECMWF's Scalable Acquisition and Pre-Processing system https://www.ecmwf.int/en/elibrary/17341-sapp-new-scalable-acquisition-and-pre-processing-system-ecmwf
SPICE	Stacked Python libraries for Improved Computing Efficiency (SPICE), is an internal project at ECMWF focused on pooling resources to refactor the Python software stack into more reusable components