ECMWF Feature article

from Newsletter Number 170 – Winter 2021/22

COMPUTING

Aviso: ECMWF's data availability notification service

www.ecmwf.int/en/about/media-centre/media-resources

doi: 10.21957/c59dk312pi

This article appeared in the Computing section of ECMWF Newsletter No. 170 – Winter 2021/22, pp. 26–31.

Aviso: ECMWF's data availability notification service

Claudio Iacopino, Tiago Quintino, James Hawkes, Baudouin Raoult (all ECWMF), Mikko Partio (FMI), Martin Grønlien Pejcoch (MET Norway), Tomas Karlsson (SMHI)

To ensure the timely exploitation of its data, ECMWF has developed a system, called Aviso, designed to notify users when real-time forecast data or derived products are available, and to trigger user-defined workflows in an automated fashion. The Aviso service has been available since the beginning of 2021 to users of the European Weather Cloud (EWC). A few ECMWF Member States have already started exploring its advantages by executing time-critical workflows directly in the EWC. This article gives an overview of the Aviso service and showcases some of these workflows. Aviso is openly available on GitHub: *https://github.com/ecmwf/aviso*.

Motivation

Recent investments in cloud-based platforms have attracted a growing number of consumers of ECMWF data. An example of these initiatives is the EWC, an ECMWF–EUMETSAT cloud infrastructure built on ECMWF premises. EWC users wish to run automated, real-time tasks or workflows close to the latest data produced by the model in the ECMWF high-performance computing facility (HPCF), thus avoiding costly data transfers out of the data centre. This trend is likely to increase with the exponential growth of weather forecast data.

The real-time forecast stream currently produces about 120 TiB of raw weather data every day, while derived products amount to about 45 TiB of disseminated data per day. It is expected that in the next few years, taking into account resolution upgrades and more complex model physics, the raw forecast data will exceed a petabyte per day.

From an operational perspective, such a large amount of data and users requires an efficient way of synchronising the users' workflows with ECMWF's forecast schedule. A mechanism is needed to timely notify the consumers of specific data availability in a scalable manner and provide the capability to automatically trigger their workflows based on this data. ECMWF's current systems allow users to access the data only in specific time periods defined by the forecast schedule. Moreover, they are not designed to provide notifications to a large number of users. Aviso allows users to synchronise their workflows directly with the availability of the data.

Overview

Aviso is a scalable notification system designed for a high throughput of notifications. It is developed by ECMWF with the following objectives:

- Notifying data availability events
- Domain-specific – speaks ECMWF meteorological language
- Triggering user workflows
- Supporting a semantic when <this> … do <that> …
- Persistent history and ability to replay events
- Independent service from HPC or cloud environments
- Protocol-agnostic
- Highly reliable and built for time-critical applications.

Aviso is developed with the intention of being generic and applicable to various domains and architectures. Notifications are designed to provide a lightweight message. ECMWF uses Aviso to provide notifications about data availability. Each notification is composed of a set of metadata required to identify and address the data availability event and a field to inform about the data location (e.g. URL). Moreover, these metadata are expressed using ECMWF Meteorological Archival and Retrieval System

(MARS) language. This provides a richer experience to the users as they can reuse the same MARS keys used to define products and access data to subscribe to events of data availability.

Aviso is a client-server application. We refer to the notification server as Aviso Server and to the client application as Aviso Client, or simply Aviso. Aviso Server is a micro-services application deployed on the ECMWF Kubernetes infrastructure. This guarantees high levels of reliability, elasticity and scalability. The solution is based on a key–value store implementation where data availability events are persisted: the key represents the event metadata, while the value contains the data location. Unlike queue-based technologies, this key–value store makes it possible for notifications to be replayed, and it enables catch-up mechanisms which are more flexible for consumers and greatly increase the reliability of the system.

The remainder of the article focuses on Aviso Client, which is more relevant to the user community.

Notification events

ECMWF has deployed Aviso as a notification service for the availability of the data produced by the Centre. Figure 1 shows the ECMWF data flow. It starts from the data assimilation of observations, followed by the generation of model output. The latter is a time-critical step relevant for users' workflows, and therefore Aviso is notified when this data is available in the HPC fast storage (MARS/Fields Database (FDB)). The data flow continues with the generation of derived products that are then disseminated via ECMWF's dissemination system. The delivery of these products to their individual destinations is also notified to Aviso as users depend on custom products for their downstream applications.

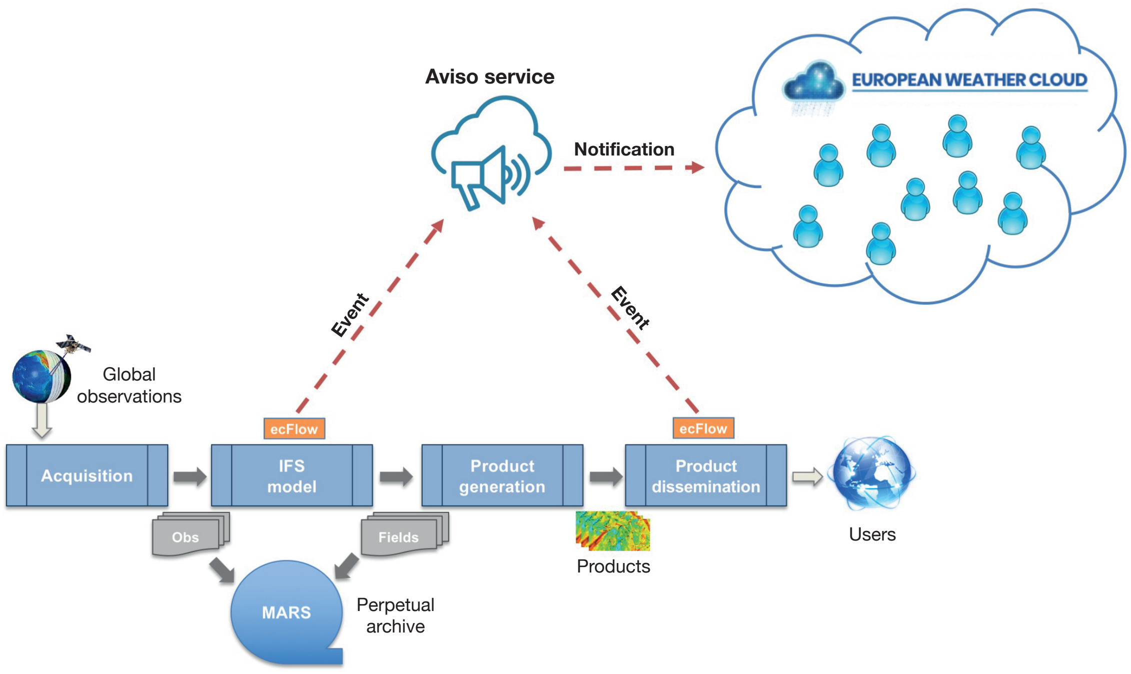


Figure 1 Events submitted to Aviso from the ECMWF data flow.

Aviso thus currently offers notifications for the following types of events:

- User products delivered via ECMWF's dissemination system. Each event of this type is related to one product disseminated to the user destination. The event contains the URL to access the product notified together with a set of metadata identifying the product.
- Availability of data in the ECMWF HPC fast storage (MARS/FDB). Each event of this type is related to one step of the forecast model output. The event carries all the metadata identifying these model step data. It does not contain its location because users will have to access it via the conventional MARS API.

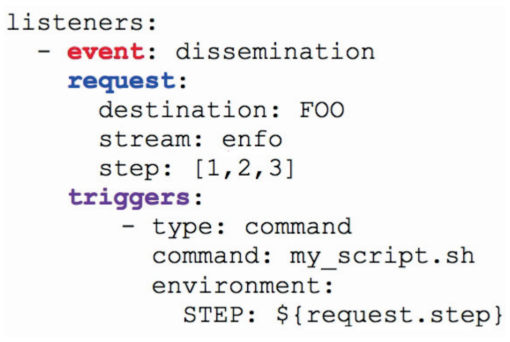
How to use Aviso

This section gives a short introduction on how to use Aviso Client. Detailed user documentation is available on Read the Docs: *https://pyaviso.readthedocs.io*

Aviso Client is a Python package available and installable from PyPI: *https://pypi.org/project/pyaviso/*. It can be used as a Command Line Application (CLI) or as a Python API, to integrate into users' own tools. In both cases users must subscribe to an event and program a trigger associated with the event. This is done through a listener configuration. A listener is composed of the following elements:

- **event** – the type of event to listen to
- **request** – a dictionary of keys identifying the specific events to be notified
- **triggers** – a sequence of processes to be executed once a notification is received

Figure 2 shows a YAML file used as an example of this configuration. This is a basic example of a listener to dissemination events, identified by the keyword dissemination. In Aviso, we use event types to distinguish the nature of the events. The currently available types of events are mars and dissemination. A mars event notifies when a new step of the forecast model output is available in the HPC fast storage (MARS/FDB). A dissemination event notifies when a user's product is delivered via ECMWF's dissemination system to the predefined user destination. The event type determines how Aviso will interpret and validate the request section.



```yaml
listeners:
  - event: dissemination
    request:
      destination: FOO
      stream: enfo
      step: [1,2,3]
    triggers:
        - type: command
          command: my_script.sh
          environment:
            STEP: ${request.step}
```

Figure 2 Aviso listener configuration defining the event to listen to and the trigger to execute.

The request section allows users to define the When condition. It contains specific keys that are a subset of the ECMWF MARS language. The MARS key describes the data to which the event is related. Aviso delivers only the notifications that match the keys defined. Users can narrow or expand this selection by adding or removing keys, respectively.

The triggers section allows users to define the Do condition. It contains a list of triggers. Each trigger defines an independent action executed every time a notification is received. A few types of triggers are available, including logging, execution of a Linux shell command as an independent process or posting the notification to an external location. The latter makes it possible to submit events in other cloud ecosystems, such as Amazon, Azure, etc., enabling workflow execution across different clouds or data centres. The notification can be sent to Amazon Simple Notification Service (SNS) topics or to HTTP endpoints implementing the CloudEvents standard. In the example, the script my_script.sh will be executed.

Using Aviso as Python API is intended for users who want to integrate it into a bigger workflow written in Python, or who simply have their trigger defined as a Python function. Figure 3 shows an example of a Python script that defines a function to be executed once a notification is received, creates a listener that references this function, and finally passes it to Aviso to execute. The function in the example, called my_mars_request, performs a MARS request to retrieve the data notified. Note how the notification metadata that Aviso provides is then used to build the MARS request. Aviso can achieve this seamless integration by virtue of being a domain-specific notification system for meteorological data.

```
from pyaviso import NotificationManager
from ecmwfapi import ECMWFService

# define function to be called
def my_mars_request(notification):
    mars_server = ECMWFService("mars")
    request = notification["request"]
    request.update({
        "type": "fc",
        "levtype": "sfc",
        "param": 167.128})
    mars_server.execute(request, "my_data.grib")

# define the trigger
my_trigger = {"type": "function", "function": my_mars_request}

# create an event listener request that uses that trigger
my_request = {"stream": "enfo", "date": 20190810, "time": 0, "step": 0}
listeners = {"listeners": [{"event": "mars",
                            "request": my_request,
                            "triggers": [my_trigger]}]}
# run it
aviso = NotificationManager()
aviso.listen(listeners=listeners)
```

Figure 3 Example of using Aviso Python API to define a function to be executed once a notification is received.

Catching up on missed notifications

Before listening to new notifications, Aviso will by default check what was the last notification delivered for that user, and it will then return all the notifications that have not yet been delivered. It will then carry on listening and deliver new ones. This behaviour ensures high reliability because, even if the listening process is interrupted due to networking faults or system unavailability, the users will not miss any notifications.

Users can also choose to listen only to new notifications or replay past notifications where available. Replaying past notifications is also a useful mechanism for testing the listener configuration with real notifications.

Pilot workflows

This section showcases three workflows implemented by Member State users in the EWC. These workflows are pilot applications. They aim to explore and demonstrate the potential in using Aviso as a notification system and triggering mechanism for downstream applications of weather and climate data that exploit the data proximity of the EWC.

MET Norway workflow

MET Norway has always been keen to make ECMWF data available as soon as possible for downstream processing and utilisation. In 2016, it developed a system broadcasting an event every time a model time step is received from the dissemination system, triggering further actions. Dissemination goes through the Secure File Transfer Protocol (SFTP), and MET Norway uses the inotify Linux subsystem to detect a change. With Aviso, this task has become easier. Aviso provides these events, and ECMWF disseminates the data via an Amazon Simple Storage Service (S3) object store in the EWC.

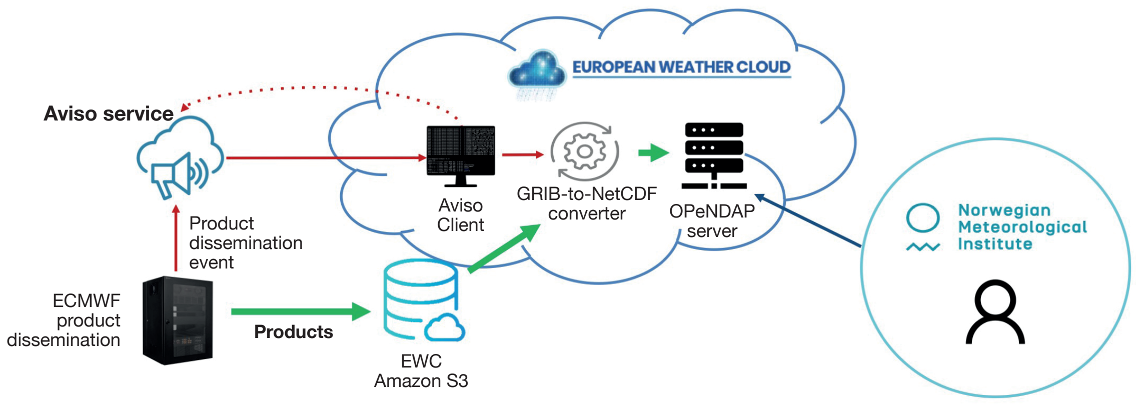


Figure 4 MET Norway workflow using Aviso with the EWC.

Figure 4 shows the new set-up. Aviso Client is used in a proof-of-concept application, which converts GRIB files to NetCDF and serves them using an OPeNDAP server (THREDDS). Systemd is the orchestrator used to run Aviso and schedule conversion jobs using a language-agnostic background job system called Faktory (*https://github.com/contribsys/faktory*). The notifications include a full S3 path, which is reachable from wherever the conversion system is running.

This set-up makes use of templating in the configuration file to structure commands for the processing system. This feature makes it possible to parametrize post-processing programs and does not pose any requirements on the languages or libraries used. The implementation of the pipeline described is available at *https://github.com/metno/agnc*.

To leverage the flexibility of S3, MET Norway is planning to add a conversion to the Zarr format and distribute the data further by pushing it back to the S3 object storage.

Finnish Meteorological Institute workflow

The Finnish Meteorological Institute (FMI) is planning to use Aviso to integrate an existing Amazon Web Services (AWS) processing infrastructure with a data processing pipeline deployed at the EWC. The overall workflow serves the following purposes:

- Converting GRIB data to a format used by an FMI Weather Information and Forecast Production System (SmartMet) workstation
- Exporting data to Geoweb as Web Map Service (WMS) layers
- Enabling product development through an interactive platform.

Figure 5 shows the overall concept of the pipeline at the EWC. A preliminary version of this pipeline is deployed as a Kubernetes cluster hosted by the EWC. Aviso is used to signal whenever a derived product has been disseminated to the FMI S3 bucket at the EWC. From this notification, a JSON message is sent to an Amazon SNS topic at the AWS cloud. By doing this, FMI is connecting the EWC pipeline with an AWS processing infrastructure previously developed; the SNS topic is part of that infrastructure. The message is then delivered back to the EWC, where a serverless function implemented by OpenFaaS is reading it and starting a subsequent processing step, dependent on the message contents.

One important task is forecast postprocessing, for example producing probability products with specific thresholds from ensembles. Another task is to convert GRIB data to a file type supported by the SmartMet workstation used by FMI meteorologists. As part of the FMI pipeline, a SmartMet server instance is running continuously at the EWC, providing WMS layers directly from GRIB data. These layers are read by a Geoweb instance running at AWS.

Another service is offered by a Dask instance running at the EWC, which provides an interactive development platform in the form of Jupyter notebooks to enable development of new products and algorithms. Resulting data is exported to the SmartMet server and visualised with Geoweb.

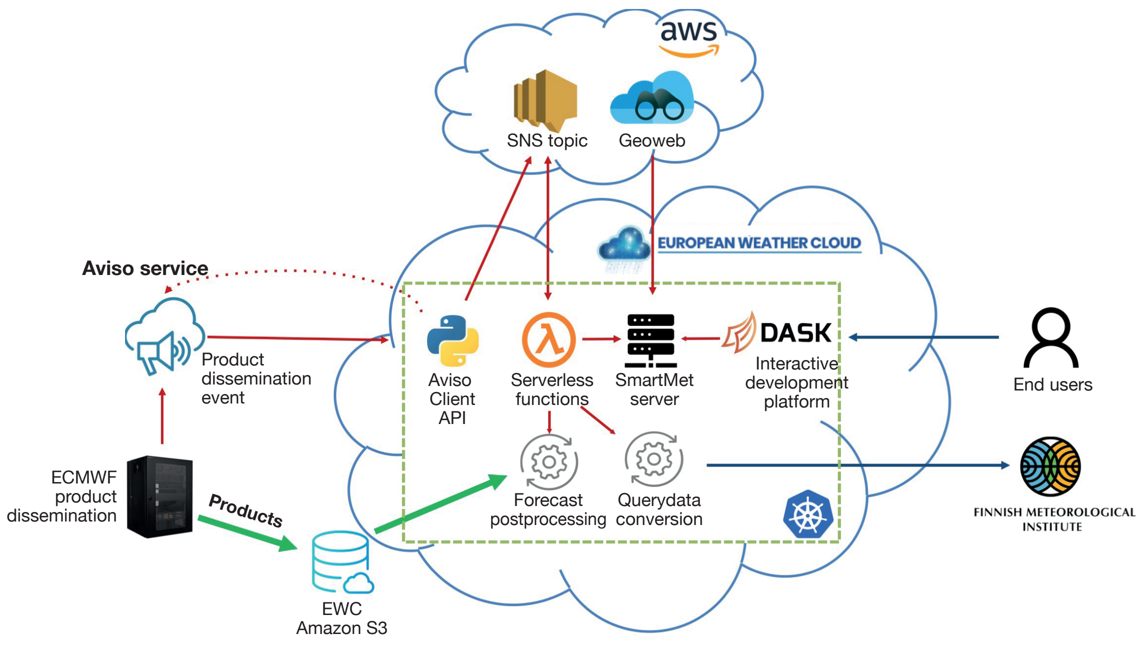


Figure 5 Finnish Meteorological Institute workflow using Aviso with the EWC.

Swedish Meteorological and Hydrological Institute workflow

The Swedish Meteorological and Hydrological Institute (SMHI) currently uses a central ecFlow workflow package to control and supervise all data production pipelines that involve the retrieval and processing of derived products from the ECMWF Production Data Store (ECPDS) and model output data from the MARS archive. In this architecture, Aviso can fill a void as currently there is no control flow support for new data available in MARS or ECPDS. At present, active polling is used to trigger the various pipelines, which puts unnecessary load on the servers. Figure 6 illustrates how Aviso would be used instead. Some preliminary tests have been conducted in running Aviso Client at SMHI and to use events from Aviso to set events in SMHI's ecFlow service. The plan is to trigger postprocessing jobs in both the EWC and in the SMHI private cloud from the central ecFlow at SMHI. This would make it possible to use the current infrastructure to control and also supervise production pipelines located in external cloud infrastructures like the EWC.

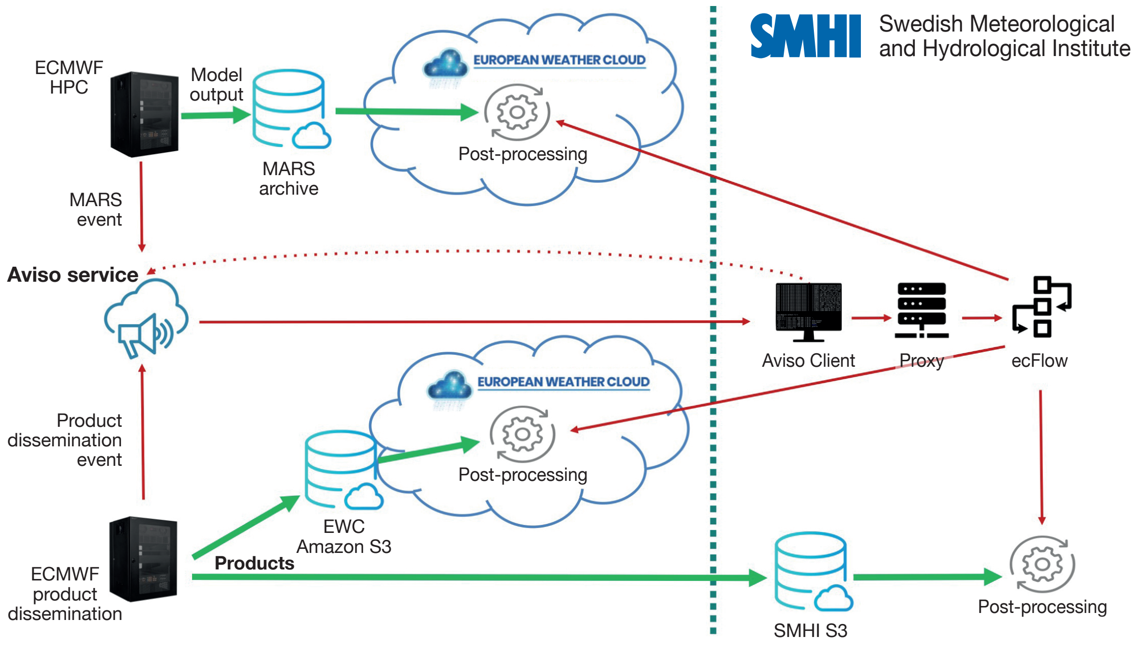


Figure 6 Swedish Meteorological and Hydrological Institute (SMHI) workflow using Aviso with the EWC.

Outlook

Aviso is currently pre-operational, and it is receiving nearly 400,000 notifications a day of both model output data and derived product availability. Users of the European Weather Cloud have started to explore the potential of this new service by building responsive data processing pipelines. Aviso is, however, not confined to the EWC infrastructure and can offer notifications across different data centres and cloud environments thanks to the adoption of cloud standards such as CloudEvents and AWS SNS topics.

Aviso is planned to acquire operational status in 2022, together with IFS Cycle 48r1. Several activities are taking place to achieve this target, including the set-up of the service support teams, lines of communication, operational procedures (including disaster recovery), training and the implementation of monitoring and analysis infrastructures. With respect to users, the intention is to gradually open the service to a larger audience while also integrating it with other ECMWF operational systems.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No. 825532 and 824115.

Further reading

Pappenberger, **F.** & **M. Palkovič**, 2020: Progress towards a European Weather Cloud, *ECMWF Newsletter* **No. 165**, 24–27.

Gougeon, **L.**, 2019: The ECMWF Production Data Store, *ECMWF Newsletter* **No. 159**, 35–40.