The Journal of Open Source Software

CFTime.jl: a Julia package for representing time following the Climate and Forecast conventions

Alexander Barth[1]

1 GHER, University of Liège, Liège, Belgium

DOI: 10.21105/joss.08487

Software

- Review
- Repository
- Archive

Editor: Mengqi Zhao

Reviewers:

- @haakon-e
- @alextbradley
- @MartinuzziFrancesco

Submitted: 11 April 2025
Published: 30 September 2025

Summary

The Climate and Forecast (CF) conventions are a metadata standard for Earth data (B. Eaton et al., 2024) and are mainly used in oceanography and meteorology. The CF conventions were originally proposed for the NetCDF storage format, but they are also increasingly used with other formats like Zarr (*Zarr Storage Specification 2.0 Community Standard*, 2022) and GRIB (`GRIBDatasets.jl`).

Since the initial release of the CF conventions (B. Eaton et al., 2003), the encoding of time instances has been standardized. The Julia package `CFTime` implements various calendars that have been standardized in the frame of these conventions. It also supports some arithmetic operations for example, computing the duration between two time instances, ordering time instances and creating time ranges. The time origin and resolution are flexible ranging from days to attoseconds.

Statement of need

In many Earth science disciplines and beyond, expressing a time instance and a duration is essential. The CF conventions provide a rich and flexible framework for handling time, equally applicable to observations and model data. To our knowledge, `CFTime.jl` is the only package in the Julia ecosystem that implements the time structures standardized by the CF conventions. While almost all datasets used in Earth science use dates after the year 1582, some datasets or software systems (e.g. J. W. Eaton et al., 2025; Lowry, 2024) use a time origin before this date, which makes it necessary to handle the transition from the Julian to the Gregorian calendar. Some users also expressed the need for microseconds and nanoseconds as time resolutions even if they are rarely used in typical Earth science applications (`CFTime` github issue 18 and `NCDatasets` github issue 248).

As of 18 September 2025, 132 Julia packages depend directly or indirectly on `CFTime` (excluding optional dependencies). For example, `CFTime` is used by numerical models, such as `ClimaOcean.jl`, a framework for realistic ocean and coupled sea-ice simulations based on `Oceananigans.jl` (Ramadhan et al., 2020), the hydrological modeling package `Wflow.jl` (Verseveld et al., 2024) and `AIBECS.jl`, a modeling framework for global marine biogeochemical cycles (Pasquier et al., 2022).

Several data-related packages also make direct or indirect use of `CFTime`, such as the NetCDF manipulation package `NCDatasets.jl` (Barth, 2024), the gridded data processing package `YAXArrays.jl` (Gans et al., 2023) and packages for handling in-situ data from various observing platforms (`OceanRobots.jl` (Forget, 2024) and `ArgoData.jl` (Forget, 2025)).

State of the field

The API of `CFTime` was highly influenced by Julia's `Dates` module from the standard library. The `Dates` module implements the `DateTime` structure representing a time instance in the proleptic Gregorian calendar following the ISO 8601 standard. The time instances are encoded using a 64-bit integer with millisecond precision and 31 December 1 BC 00:00:00 as the starting date. In the Python ecosystem, the calendars from the CF conventions are implemented by the `cftime` package. The Python package implements the time to microsecond accuracy (as of version 1.6.4). A timestamp is represented internally by storing separately the year, month, day, hour, minute, second and microsecond. This reduces the risks of integer overflows at the expense of memory requirements and complexity of performing arithmetic operations.

Installation

`CFTime` supports Julia 1.6, and later and can be installed with the Julia package manager using the following command:

```julia
using Pkg
Pkg.add("CFTime")
```

`CFTime` is a pure Julia package and currently depends only on the modules `Dates` and `Printf`, which are part of Julia's standard library.

Features

In the context of the CF conventions, a time instance is represented as a time offset measured from a time origin (in UTC): for example, the value 86400 with units "seconds since 1970-01-01 00:00:00" is 2 January 1970, 00:00:00 UTC. The units of the time offset and the time origin are stored in the `units` attribute of the time variable.

The `calendar` attribute of a NetCDF or Zarr time variable defines how the time offset and units are interpreted to derive the calendar year, month, day, hour, and so on. The CF conventions define several calendar types, including:

Calendar	Type	Explanation
`standard`, `gregorian`	`DateTimeStandard`	the Gregorian calendar after 15 October 1582 and the Julian calendar before
`proleptic_gregorian`	`DateTimeProlepticGregorian`	the Gregorian calendar applied to all dates
`julian`	`DateTimeJulian`	the Julian calendar applied to all dates
`noleap`, `365_day`	`DateTimeNoLeap`	calendar without leap years
`all_leap`, `366_day`	`DateTimeAllLeap`	calendar with only leap years
`360_day`	`DateTime360Day`	calendar assuming that all months have 30 days

`CFTime` is based on the Meeus' algorithm (Meeus, 1998) for the Gregorian and Julian calendars, with two adaptations:

- The original algorithm is based on floating-point arithmetic. The algorithm in `CFTime` is implemented using integer arithmetic, which is more efficient. Additionally, underflows and overflows are easier to predict and handle with integer arithmetic.
- The Meeus' algorithm has been extended to dates prior to 100 BC.

The following is a list of the main features of `CFTime`:

- Basic arithmetic, such as subtracting two time instances to compute their duration, or adding a duration to a time instance.
- Support for a wide range of time resolutions, from days down to attoseconds, for feature parity with NumPy's `datetime64` type (Harris et al., 2020; *NumPy API Reference*, 2024).
- Support for arbitrary time origins. Since the time origin for NumPy's `datetime64` type is fixed to 1 January 1970 at 00:00, the usefulness of some time units is limited. As an extreme example, with attoseconds, only a time span of ± 9.2 s around the time origin can be represented since a 64-bit integer is used internally.
- By default, the time counter is a 64-bit integer, but other integer types or floating-point types can be used.

Acknowledgements

I thank all contributors to this package, in particular Martijn Visser, Fabian Gans, Haakon Ludvig Langeland Ervik, Rafael Schouten and Yeesian Ng. I also acknowledge Jeff Whitaker for Python's cftime which has helped the development of this package by providing reference values and a reference implementation for tests.

Funding

The author acknowledges the F.R.S.-FNRS (Fonds de la Recherche Scientifique de Belgique) for funding his position. This work was partly performed with funding from the Blue-Cloud 2026 project under the Horizon Europe programme, Grant Agreement No. 101094227.

References

Barth, A. (2024). NCDatasets.jl: a Julia package for manipulating netCDF data sets. *Journal of Open Source Software*, *9*(97), 6504. https://doi.org/10.21105/joss.06504

Eaton, B., Gregory, J., Drach, B., Taylor, K., & Hankin, S. (2003). *NetCDF Climate and Forecast (CF) Metadata Conventions v1.0*. CF Conventions Committee. https://cfconventions.org/Data/cf-conventions/cf-conventions-1.0/build/cf-conventions.html

Eaton, B., Gregory, J., Drach, B., Taylor, K., Hankin, S., Caron, J., Signell, R., Bentley, P., Rappa, G., Höck, H., Pamment, A., Juckes, M., Raspaud, M., Blower, J., Horne, R., Whiteaker, T., Blodgett, D., Zender, C., Lee, D., … Bartholomew, S. L. (2024). *NetCDF Climate and Forecast (CF) Metadata Conventions* (Version 1.12). CF Community. https://doi.org/10.5281/zenodo.14275599

Eaton, J. W., Bateman, D., Hauberg, S., & Wehbring, R. (2025). *GNU Octave version 10.1.0 manual: A high-level interactive language for numerical computations*. https://www.gnu.org/software/octave/doc/v10.1.0/

Forget, G. (2024). Digital Twins for Ocean Robots. *Proceedings of the JuliaCon Conferences*, *6*(65), 164. https://doi.org/10.21105/jcon.00164

Forget, G. (2025). *euroargodev/ArgoData.jl: v0.2.0* (Version v0.2.0). Zenodo. https://doi.org/10.5281/zenodo.14788589

Gans, F., Cremer, F., Alonso, L., Kraemer, G., Dimens, P. V., Gutwin, M., Pabon-Moreno, D. E., Kong, D., Martin, Martinuzzi, F., Chettouh, M. A., Loos, D., Zehner, M., Roy, P., Zhang, Q., ckrich, Glaser, F., & linamaes. (2023). *JuliaDataCubes/YAXArrays.jl: v0.5.2* (Version v0.5.2). Zenodo. https://doi.org/10.5281/zenodo.8414000

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Lowry, M. A. S., Roy AND Fichaut. (2024). *SeaDataNet. Datafile formats. ODV, MEDATLAS, NETCDF* (SeaDataNet, Ed.) [Report]. https://doi.org/10.13155/56547

Meeus, J. H. (1998). *Astronomical algorithms*. Willmann-Bell, USA. ISBN: 0943396611

NumPy API reference: Datetimes and timedeltas. (2024). https://numpy.org/doc/stable/reference/arrays.datetime.html

Pasquier, B., Primeau, F. W., & John, S. G. (2022). AIBECS.jl: A tool for exploring global marine biogeochemical cycles. *Journal of Open Source Software*, *7*(69), 3814. https://doi.org/10.21105/joss.03814

Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza, A., Edelman, A., Ferrari, R., & Marshall, J. (2020). Oceananigans.jl: Fast and friendly geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, *5*(53), 2018. https://doi.org/10.21105/joss.02018

Verseveld, W. J. van, Weerts, A. H., Visser, M., Buitink, J., Imhoff, R. O., Boisgontier, H., Bouaziz, L., Eilander, D., Hegnauer, M., Velden, C. ten, & Russell, B. (2024). Wflow_sbm v0.7.3, a spatially distributed hydrological model: From global data to local applications. *Geoscientific Model Development*, *17*(8), 3199–3234. https://doi.org/10.5194/gmd-17-3199-2024

Zarr storage specification 2.0 community standard (No. 21-050r1). (2022). Open Geospatial Consortium. https://doi.org/10.62973/21-050r1