WMO Guide to Free and Open Source Software

World Meteorological Organization

Date: 2026-02-11

Version: 0.1.0

Document status: DRAFT

Document location: https://wmo-im.github.io/wmo-foss-guide/guide/wmo-foss-guide-DRAFT.html

WMO publication location: TBD

Standing Committee on Information Management and Technology (SC-IMT)[1]

Commission for Observation, Infrastructure and Information Systems (INFCOM)[2]

Table of Contents

Introduction

Open source software has become increasingly crucial for WMO and its Members. No longer a "niche" industry, FOSS now powers major initiatives and services worldwide, and is backed by industry support.

FOSS solutions play a pivotal role as accelerators for the implementation of Early Warnings for All, a key WMO initiative aimed at protecting every person on Earth with life-saving early warning systems by 2027. By providing accessible, customizable, and cost-effective tools, FOSS enables Members to rapidly deploy and adapt early warning systems to their specific needs and contexts. This approach is particularly crucial for developing countries (LDCs) and small island developing states (SIDS), where resource constraints often hinder the implementation of proprietary solutions.

Moreover, FOSS initiatives serve as powerful catalysts in supporting Members' efforts towards digital transformation. As National Meteorological and Hydrological Services (NMHSs) worldwide strive to modernize their operations and services, FOSS now offer a flexible and scalable foundation for innovation. They enable Members to leverage cutting-edge technologies, collaborate on development, and share best practices, thus accelerating their digital transformation journeys while optimizing resource utilization.

Audience

The document is intended for WMO Members who are involved in software development activities in their respective organizations (developers, managers/decision makers) as well as WMO Secretariat in the coordination, alignment and support of FOSS within WMO.

Scope

This document provides guidance on FOSS for WMO Members and WMO Secretariat.

Background

- use notes from TT-OSS document to INFCOM Management Group
- strong usage, increasing usage
- WIS 2.0 as an example of FOSS dev during standards dev
- needs coordination

FOSS development and use among WMO Members has been a longstanding activity. Typical FOSS implementations included decoders and encoder libraries of WMO BUFR and GRIB formats (e.g. ecCodes, degrib, libecbufr, etc.), as well as data dissemination services (THREDDS, GDAL, ERDDAP, etc.).

In addition, as part of the global open data/open science movement, numerous government organizations have put forth using FOSS by default; this includes use of exiting FOSS tools as well as developing of same, working in the open by default. Free software development platforms and tools (for example, GitHub) have greatly lowered the barrier for sharing software for research,

development and operations for all.

As part of the development of the WMO Information System (WIS2), FOSS implementation has been a significant activity, as exemplified by a number of Reference Implementations of WIS2 Standards (such as wis2box, wis2-gdc, etc.). The development of software in the open during the pre-operational phases of WIS2 proved valuable in testing WIS2 Standards put forth in Technical Regulations while in development, promoting the “release early, release often” philosophy of agile and iterative development. While FOSS is resident in numerous WMO Member organizations, guidance and coordination becomes more vital to ensure Member services are provided with security, privacy and safety in mind for WMO Members and beyond.

Data policy considerations (TODO Athina)

Enabling Unified Data Policy via software

Open data policies

FAIR data principles

[1] https://community.wmo.int/governance/commission-membership/commission-observation-infrastructures-and-information-systems-infcom/commission-infrastructure-officers/infcom-management-group/standing-committee-information-management-and-technology-sc-imt

[2] https://community.wmo.int/governance/commission-membership/infcom

Guidelines

WMO Members

Using FOSS

- FOSS as an option during software evaluation
- Risk, hidden costs (TODO Jian)
  - Principles apply to ANY software
  - Risk management
  - Due diligence (maintenance, updates)
- Lifecycle management/EOL → migration
- Total cost of ownership considerations
  - HR profile / IT capacity of organization
- Benefits (freedom, cost, reducing vendor lock in, portability) (TODO Jian)
- Infrastructure considerations

Risks

1. **License compliance risks** Open source licenses often have widely varying requirements regarding commercial use, redistribution, and code modification. Certain licenses include restrictive clauses, such as mandating that derivative works be made open source or that original copyright notices be preserved in full. Violations may lead to legal disputes and intellectual property issues.
2. **Security and continuity risks**: Open source software from niche projects or less-established projects may pose security risks due to limited formal code audits. Additionally, the complexity of open source software supply chains creates vulnerabilities that could be exploited to implant malicious code. These risks are amplified in meteorological applications, especially when handling sensitive observational data or real-time forecasting models.
3. **Maintenance risks**: Fixes for vulnerabilities rely on community responsiveness. If an open source project is discontinued, known vulnerabilities may remain unaddressed.
4. **Support risks**: Most open source software lacks official support. As a result, timely response and effective resolutions of critical operational system failures cannot be guaranteed, potentially leading to interruptions in meteorological operation and data loss.

Hidden costs

1. **Technical adaptation and integration costs**: Human resources must be allocated for secondary development (customization), system interface integration, and compatibility testing to ensure the software aligns with existing business systems.
2. **Long-term maintenance and version management costs**: Dedicated personnel are needed to track version updates, conduct testing, and manage migrations. If a project is discontinued,

taking over its maintenance internally entails high technical challenges and added operational complexity.

3. **Talent acquisition and training costs**: Recruiting skilled professionals can be expensive, and ongoing in-house training is required to maintain the necessary expertise.
4. **Compliance audit and management costs**: It is necessary to maintain an inventory of all open source components and perform regular compliance audits. Large meteorological organizations may also need to develop dedicated management platforms, with costs increasing as the number of components grows.

Contributing to FOSS

National policies

The adoption and development of FOSS within NMHSs can be significantly strengthened through supportive national-level policies. Several WMO Members have already introduced digital government strategies that encourage the use of open standards, open data and open-source software as default options within public administrations. Where such frameworks exist, NMHSs can leverage them to increase efficiency, reduce vendor lock-in and ensure long-term sustainability of operational systems.

National FOSS-related policies may include:

- "Open by default" procurement rules requiring public institutions to consider FOSS alternatives alongside proprietary solutions when acquiring software. In some countries, like Romania, when you purchase "on demand" software development services, the source code should be transferred to the customer along with all the IPR associated with the software. Making the source code available under an open-source license is highly recommended.
- Guidelines for software development funded from public budget, encouraging publication of code under approved open-source licenses.
- Policies promoting interoperability and open standards, which naturally align with FOSS implementations within WIS2 and other WMO technical frameworks.
- Capacity-building programs that enable staff to acquire skills for maintaining and developing open-source tools, reducing long-term operational dependency on external vendors.
- Security and lifecycle requirements to ensure that open-source components used in critical systems follow maintenance, patching, auditing and version management rules.

Where no national regulations exist, NMHSs can still develop institution-level policies or internal guidelines to support responsible FOSS adoption:

- Establish internal rules for licensing, contribution approval and repository management.
- Define procedures for evaluating FOSS components (governance, community health, maintenance model).
- Incorporate sustainability planning (including contribution back to upstream projects when fixes or new features are developed internally).
- Promote transparency by adopting open repositories for research software and operational

tools.

- Events/hackathons (e.g. OGC/OSGeo/ASF Joint Sprints) TODO Vasile
  - By product: connection/collab
- Regulations / risk / constraints / considerations
  - Features, maintenance, project

Managing FOSS activities

In some cases, Members developing their own FOSS in support of WMO obligations and activities may wish to make the software freely available to the public. The current FOSS landscape and infrastructure provides a strong support mechanism for doing FOSS development in the open, using services such as GitHub, GitLab, Codeberg and numerous other platforms.

In the same spirit as contributing to FOSS, where no national regulations exist, NMHSs can still develop institution-level policies or guidelines in support of making FOSS available to the public. The following are (some) key aspects to consider:

- Establish an "architecture of participation":
  - Who will be responsible for development, release management, and accepting external contributions?
  - How will commit access be governed and managed? Does the project need a project steering committee or group of stakeholders to help drive the project?
  - How will external collaboration be addressed? For example, will the project have a bug or issue tracker which can be used to identify new feature development, defects and/or bugs. Will there be a dedicated discussion forum or mailing list for wider discussion?
  - Make clear how collaboration will work to the public (for example, by providing a `CONTRIBUTING.md` document for GitHub)
  - Make clear how security issues should be addressed (mitigation/release policies/timelines, contact points, etc.)
  - What is the release schedule for new versions?
- Choose a license that the software will be made available with (such as the Apache License, Version 2.0, MIT, etc.). It is strongly advised to consider and consult organization policies to ensure proper licensing and understanding risk, responsibility and liability
- Choose the mechanism by which the software will be made available (GitHub, GtiLab, Codeberg, organizational website, etc.)
- Ensure that a `README` is available with the code stating a clear mission and purpose
- Identify any supported WMO Technical Regulations
- Ensure that the project has proper documentation for users (downloading, installing, configuring, running), as well as additinal documentation for developer or othe audience types (administrators, etc.)
- Ensure the code is versioned (for example using Git as a version control system)

Core Benefits of Open Source Software

The core benefits of open source software can be summarized on three dimensions: autonomy and control, cost optimization, and ecosystem flexibility—offering both practical advantages and adaptability. Details are as follows:

1. **Exceptional autonomy and control**: The code is open and licenses are flexible, allowing meteorological organizations to perform deep customization as needed, directly modifying the code to add or remove functional modules to suit unique operational scenarios (e.g., regional weather forecasting adjustments). This level of access also allows them to independently identify and fix vulnerabilities, optimize performance for massive meteorological data processing, and control the technical roadmap of core systems. Certain permissive licenses (e.g., MIT License, Apache License 2.0) even permit using modified code as closed source in commercial products, balancing customization needs with operational requirements.
2. **Significant cost advantages**: No software license fees or subscription fees are required, suitable for national meteorological services as well as deployments in regional forecasting centers. The availability of source code and community documentation reduces trial-and-error costs during secondary development, making necessary human resource investment more manageable. Meteorological organizations can also choose free community support or affordable third-party services (specialized in meteorological applications) instead of expensive vendor support packages.
3. **Avoiding vendor lock-in risks**: Free from proprietary interfaces and data format dependencies, enabling free choice of deployment environments (e.g., private or public clouds). Even if the original project is discontinued, maintenance can be taken over independently or by a third party. Moreover, multiple service providers are often support the same open source software, strengthening the user's bargaining power.
4. **Excellent cross-platform portability**: Most open source tools run on multiple operating systems (e.g., Windows and Linux), eliminating the need for separate purchases. Adherence to common technical standards (e.g., WMO GRIB data formats) lowers system integration difficulty. Modular design facilitates containerized deployment and cross-cluster migration, adapting to architectural upgrades in meteorological data centers.

Infrastructure considerations

WMO Activities

Coordination, alignment and support (TODO Dave/Enrico)

- Coordination/support functions
- Software selection for WMO projects and application development
- Managing FOSS activities
- Aligning with WMO ecosystem of activities
- Ensuring sustainability of FOSS usage
- Managing risk

- Functions
- People

Standards compliance

- Compatability / compliance matrix
- Open Standards <→ FOSS support matrix
- Implementation of WMO Tech Regs / compliance ?

FOSS can also play a significant role as an indicator of the feasibility of Technical Regulation implementation. As part of the standards development phase, developing the associated FOSS implementation(s) aids in determining "fit for purpose" of the specification. In addition, FOSS implementation during standards development allows for an open review and assessment on how a standard is implemented (supported network protocols, avaiability of tooling/libraries for integration, as well as speed of implementation (if a FOSS implementation during the development of a standard takes a considerable level of effort, why? Is the complexity acceptable for the scope of the standard? Determining these aspects earlier can help improve and adjust the standard while in development (as opposed to formal amendment processes once approved by WMO). The agile and parallel development of WIS2 standards and FOSS is a key example in this context (see Case study: WIS2, FOSS, and agile development).

Another indicator is the number of implementations. For example, the Open Geospatial Consortium (OGC) mandates that a proposed standard requires at least 3 implementations for consideration by its Technical Committee. This provides "proof" to the Technical Committee that the standard can be implemented, and has investment from various projects as early adopters.

In summary, while FOSS is not in scope for being identified as part of a Technical Regulation, it is a strong indicator of the the maturity, applicability and sustainability of a given standard. Implementing FOSS during standards development provides safety in investment, iterative improvements, and confidence for widespread and sustainable adoption and implementation by Members.

Software review and evaluation

- Software identification and selection (TODO Vasile)
  - Project checklist/assessment
- "Approved projects" and/or Reference Implementations
  - Make Tech Regs more concrete
  - Tech Regs → FOSS implementations
  - Should FOSS be cited in WMO Tech Regs (suggest no)
  - Criteria needed (TODO Jian)
    - Compliance (data exchange)
    - Software evaluation (FOSS!) checklist → confidence
    - Readiness

        - Bus/retirement factor
    - Rolling review
- Harmonization: regular review of ecosystem to ensure alignment and optimal use of resources

Compliance (data exchange)

- **Standards alignment**: The software adheres to WMO technical standards and common industry specifications, unifying data exchange interfaces and formats.
- **Data protection**: It complies with international conventions and national data protection regulations by implementing data desensitization, encrypted data transmission, and secure storage of sensitive meteorological data.
- **Cross-border data security**: It meets data cross-border transfer requirements in operational regions by establishing a full lifecycle data-security management mechanism. This mechanism covers data collection, processing, sharing, and archiving in line with WMO data policies.

Software evaluation

- **Core Evaluation Checklist**:
    - *Technical Quality*: Functionality completeness and accuracy for meteorological use cases, performance when processing massive meteorological datasets (e.g., real-time observational data, forecast model outputs), reliability under peak load conditions, security, maintainability, and compatibility with existing meteorological systems.
    - *Compliance*: License is OSI-approved with no license conflicts, and complies with WMO data policies and relevant regulations.
    - *Community Sustainability*: Recent community activity (within the last 6–12 months), stability of the maintenance team, regularity of version releases, and maturity of the ecosystem.
    - *Cost-effectiveness*: Controllable costs across the software lifecycle (deployment, operation, maintenance, training, migration, and decommissioning).
    - *Business Fit*: Aligns with existing meteorological operational processes, good usability for forecasters and data analysts, and support for future expansion needs.
- **Confidence Assurance:**
    - Adopt standardized evaluation processes and quantifiable metrics to ensure objectivity and repeatability of results.
    - Validate core functionality through a Proof-of-Concept (PoC) approach, incorporating feedback collected from real-world operational scenarios.
    - Conduct multi-dimensional verification, including code audits, security vulnerability scans, and in-depth open source community research, to confirm the reliability of the software.

Readiness

Completed drafting of requirements specification documents, and form a professional evaluation team (including technical, business, legal, and operations personnel).

Set up a simulated meteorological operations testing environment, and prepare test data (including

extreme weather scenario data) and necessary evaluation tools.

Define clear priorities for software requirements and establish final decision criteria, such as approval for full adoption, conditional adoption, or rejection. Document the rationale for all decisions for audit purposes.

Bus/retirement factor

Business Fit and Retirement Considerations

- Business Fit
  - Offers seamless integration with existing meteorological operational processes, minimizing the need for process adjustments.
  - Supports custom development and feature expansion to meet evolving needs (such as business scaling and new scenario integration).
  - Aligns with users' habits in meteorological operations by providing a user-friendly interface and flexible configuration options.
- Retirement Planning
  - Establish a clear data archiving plan to ensure secure storage and traceability of historical meteorological data after the software is decommissioned.
  - Proactively assess the costs of migrating to new systems, and establish a smooth migration path to ensure business continuity.
  - Define clear responsibilities and standardized procedures for software retirement to mitigate potential risks (e.g., data leakage, system conflicts, and service disruptions).

Case study: WIS2, FOSS, and agile development

The WMO Information System 2.0 (WIS 2.0) is the framework for WMO data sharing in the 21st century for all WMO domains and disciplines. It supports the WMO Unified Data policy and the Global Basic Observing Network (GBON) and makes international, regional, and national data sharing simple, effective, and inexpensive. The idea that no Member should be left behind and the objective of lowering the barrier to adoption has been at the core of WIS 2.0 development. These objectives inspire the principles underpinning the WIS 2.0 technical framework, such as adopting open standards and Web technologies to facilitate sharing of increasing variety and volume of real-time data.[1]

The WMO Executive Council, through Resolution 34 (EC-76) - Implementation plan update of the WMO Information System 2.0, endorsed the WMO Information System 2.0 (WIS2) implementation plan. The Resolution also recognized the importance of establishing a pilot phase to develop the WIS2 infrastructure and begin testing it, in order to be ready for a pre-operational phase in 2024, and then for the transition starting in 2025.

The pilot phase was completed at the end of 2023, with several Members collaborating in building the WIS2 infrastructure. Each Member had a different role in the WIS2 framework and implemented a specific component. Starting in January 2024, the implementation of WIS2 entered the pre-operational phase in preparation for transitioning to WIS2. On 01 January 2025, WIS2

became operational.[2]

WIS2 is comprised of a number of key standards, including (but are not limited to):

- Internet and messaging protocols for data and metadata transmission (HTTP, MQTT)
- metadata encodings for dataset descriptions (WMO Core Metadata Profile) and notifications (WIS2 Notification Message)
- topic structures for real-time notifications (WIS2 Topic Hierarchy)
- interfaces and APIs for Global Services (Global Discovery Catalogue)

Throughout the WIS2 Implementation timeline, the architecture, standards definitions as well as key software components were developed, deployed and tested in parallel, in an agile manner. These software include (but are not limited to):

Software	**WIS2 component(s)**
wis2box	WIS2 Node
wis2-gdc	Global Discovery Catalogue
csv2bufr	CSV to BUFR encoding tool for station data
wis2downloader	Python package for downloading real-time data from WIS2
pywis-pubsub	Python package providing notification message publishing, subscription and data download capability in WIS2

Driven by collaboration, the development and testing of FOSS significantly helped with early assessment of the WIS2 standards in development. As a result, WIS2 standards were being tested in an agile environment and rapid feedback was provided that helped refine, adjust and improve the standards early and often.

WIS2 FOSS implementations have resulted in a more stable and robust program delivery with significantly reduced risk to Members. In addition, some tools have emerged as "Reference Implementations" as key resources for the precise implementation of various WIS2 standards which can either be deployed or studied by Members accordingly. WIS2 Technical Regulations were approved with confidence as a result of the agile development of Open Standards and Open Source.

[1] https://wmo.int/wis-20

[2] https://wmo-im.github.io/wis2-transition-guide/transition-guide/wis2-transition-guide-APPROVED.html#_1_introduction

References

Annex A: FOSS evaluation rubric

Category	Indicator
License Compatibility	License is permissive (MIT, BSD, Apache 2.0)
	Compatible with NMHS mandates and data policies
	No restrictions on data sharing or commercial use
	License is OSI-approved
	License is clearly stated (no license = reject)
Security	Clear security vulnerability policy/process
	Prompt patch releases (within days)
	No known unpatched CVEs
	Regular security checks (e.g., daily or automated)
Project Health	Public issue tracker
	Source code under version control (e.g. on GitHub, GitLab, Codeberg or other version control system)
	Active communication channels (forum, mailing list, chat)
	Wiki or documentation site
	Contribution guidelines or agreements
Community Health	Multiple active contributors
	Supported by major NMHSs, WMO Programmes, or recognized consortia
	Low bus factor (not reliant on one person)
	Code of conduct available
Openness & Governance	Transparent governance model
	Open contribution process
	Alignment with WMO standards (BUFR, GRIB2, NetCDF)
	Decision-making process documented
	Source code openly available (mandatory)
Technical Quality & Fit	High test coverage
	Continuous Integration (CI) in place
	Comprehensive documentation (user, developer, admin)
	Proven ability to handle NMHS-scale data volumes
	Validated in operational NMHS environments
	Clear release lifecycle and package management

Annex B: Examples of FOSS managed / offered by Member organizations

Organization	Platform	URL	Projects(s)
Environment and Climate Change Canada, Meteorological Service of Canada	GitHub	https://github.com/ECCC-MSC	• https://github.com/ECCC-MSC/msc-pygeoapi • https://github.com/ECCC-MSC/libecbufr • https://github.com/ECCC-MSC/msc-wis2node

Organization	Platform	URL	Projects(s)
NSF Unidata	GitHub	https://github.com/Unidata	• TODO1 • TODO2