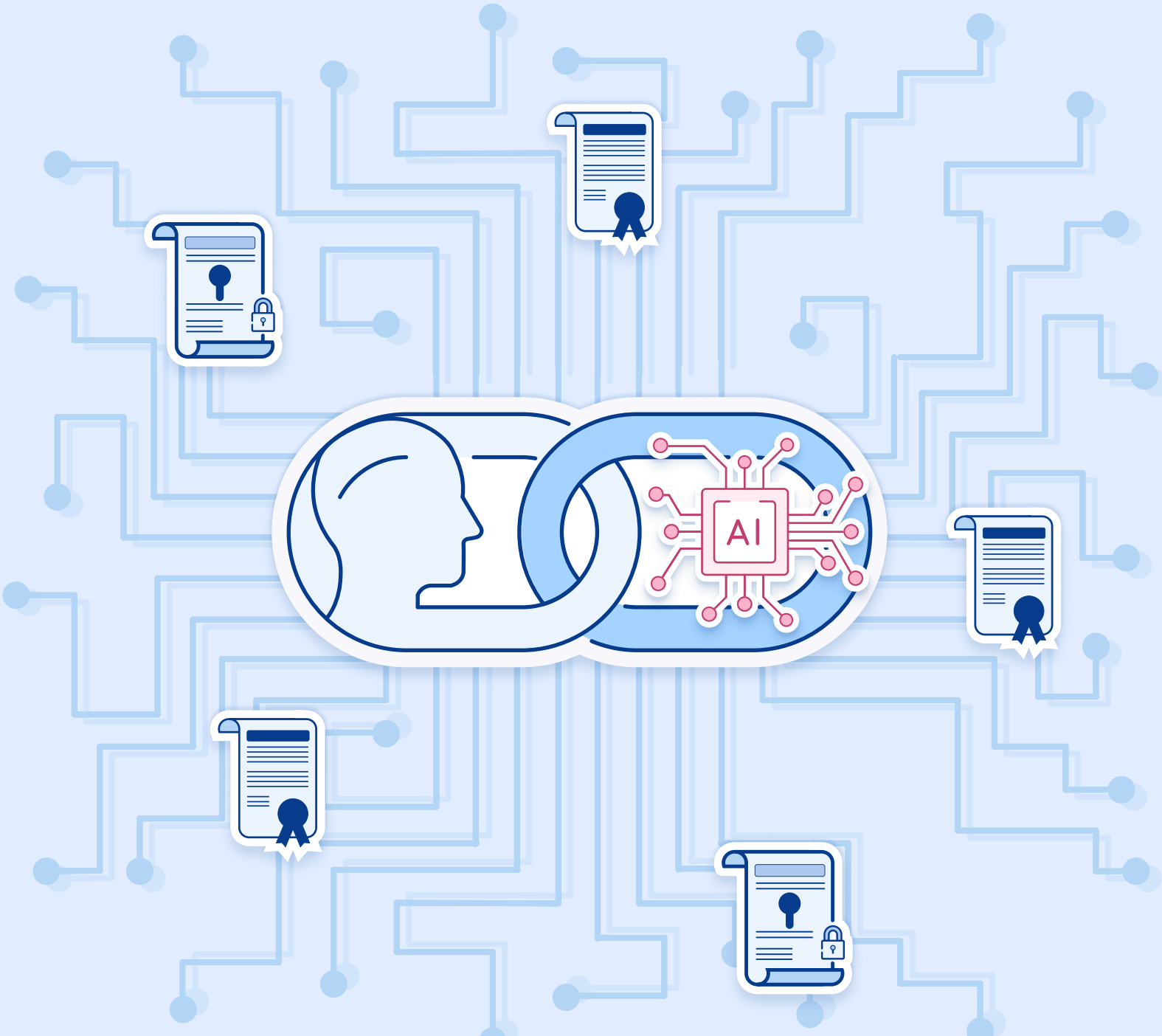
Mend.io

WHITEPAPER

The Complete Guide to Open Source & AI Licensing 2026



01

Introduction

The modern technology stack is no longer just a collection of libraries; it is a collaborative ecosystem of code and intelligence. As we navigate 2026, software engineers increasingly work alongside AI agents, creating code that is often a hybrid of human ingenuity and machine-generated logic.

While the culture of open sharing remains, the legal landscape has broadened. Open source licensing used to center on "copyleft vs. permissive." Today, teams also have to manage training-data rights, model weights, dataset terms, acceptable-use restrictions, provenance, and the legal status of AI-generated contributions. Compliance can no longer be an afterthought—it is a foundational requirement for building trustworthy, scalable, and legally sound products in the age of agentic software development.

This 2026 guide is your one-stop resource for navigating the intersection of traditional open source and the new frontier of AI licensing. It is designed to help engineering, security, legal, and product teams make fast, defensible decisions without slowing down innovation.

What this guide covers:

You'll learn how licensing and compliance applies to:

- Traditional open source code (libraries, frameworks, runtimes, build tools)
- AI model artifacts (training code, inference code, weights/parameters, evaluation assets, and model packages)
- Data-related obligations (dataset licenses, data terms, and "data information" disclosures)
- Downstream use models (fine-tuning, embedding, redistribution, container shipping, and API/SaaS delivery)

02

Open Source & AI Licensing 101

Software is protected primarily by copyright, but AI systems are assembled from multiple artifact types—each with different legal triggers and obligations. In 2026, "what's licensed" is rarely just a repository. It's usually a bundle of:

- Source code (training/inference/runtime tools)
- Model parameters (weights) and checkpoints
- Model architecture definitions
- Training and evaluation pipelines
- Data assets, data licenses, and data terms
- Documentation and notices (LICENSE, NOTICE, third-party attributions)

Open Source AI vs. Open Weights vs. Source-Available

In classic open source, a license grants the freedom to use, study, modify, and share the software under defined terms. In AI, the same idea exists, but the industry uses inconsistent labels. In practice, you'll see three categories:

1) Open Source AI (OSI definition)

The Open Source Initiative (OSI) has published an Open Source AI Definition (OSAID) that adapts open source principles to AI systems. Under this definition, an Open Source AI system must grant the freedoms to:

- Use the system for any purpose
- Study how it works and inspect its components
- Modify it for any purpose (including changing outputs)
- Share it for others to use, with or without modifications

A key precondition is access to the preferred form to make modifications, which for machine-learning systems must include:

- Data Information: sufficiently detailed information about the data used to train the system, so a skilled person can build a substantially equivalent system

Parameters: the model parameters (weights or equivalent), available under OSI-approved terms.

Important:

This is a higher bar than many "open model" releases meet in practice.

2) Open Weights

"Open weights" generally means the model's weights/parameters are available, often with inference code, but without the full ability to reproduce the system (for example, missing data information, missing training recipe details, or missing rights). Open weights releases frequently include custom terms that look more like product licenses than classic open source licenses.

3) Source-Available

Source-available licenses allow you to view source code and sometimes use it with restrictions (field-of-use restrictions, commercial thresholds, competitive use limits). These are not generally considered open source under OSI principles.

AI-generated code does not grant unlimited rights

Using an AI tool to produce code does not automatically grant you the right to use any content that appears in outputs. Your obligations depend on:

- What the AI output contains (original code, copied snippets, licensed text, etc.)
- How you use it (internal prototype vs. shipped product)
- Whether a license applies to the source material or embedded dependencies
- Your organization's policies for attribution, provenance, and security review

Also note:

some communities and companies require process disclosures (for example, tagging AI-assisted contributions). This is usually a governance rule or contribution policy—not a default requirement in OSI open source licenses.

03

Types of Licenses: Permissive, Copyleft, and AI-Specific

License categories have evolved to address how code is used and how intelligence is shared. In 2026, it's not enough to know "permissive vs. copyleft." You also need to understand how obligations are triggered by **distribution, network access, and artifact boundaries**.

Permissive Licenses (MIT, Apache 2.0, BSD)

Permissive licenses remain popular for commercial software and enterprise AI because they:

- Allow use, modification, and redistribution with minimal reciprocity
- Support proprietary layers on top of open source foundations
- Reduce friction in SaaS delivery, embedding, and commercial distribution

Practical takeaway:

permissive licenses still require compliance—especially around including license texts, preserving notices, and meeting attribution requirements.

Strong Copyleft (GPL, AGPL)

Strong copyleft licenses impose reciprocity obligations when you distribute a derivative work of the licensed code. In 2026, the most important question is still: What is being distributed, and to whom?

- **GPL**: primarily triggers on distribution (shipping binaries, distributing source, bundling in images/firmware, etc.)
- **AGPL**: closes the "SaaS loophole" by treating network interaction as a trigger in many cases.

AI-specific reality check: You will often hear sweeping claims about training on copyleft code or outputs "triggering" copyleft. In practice, the legal situation is more nuanced. The highest-confidence risk areas are:

- Shipping/copackaging copyleft components into production runtime
- Copying copyleft code into proprietary source (including "generated code" that closely mirrors protected expression)
- Failing to provide required source/offer obligations when triggers apply
- Providing AGPL software as a network service without meeting source-availability obligations for users

Treat edge cases (training data inclusion, model weights, output similarity) as high-risk legal territory that requires careful policy and counsel review. Do not assume the obligation model is identical to classic linking/distribution patterns.

Weak Copyleft (LGPL, CDDL, EPL, MPL)

Weak copyleft aims to preserve openness at a boundary:

- **LGPL**: often focused on library linking; typically allows proprietary applications to link under conditions
- **CDDL**: file-level copyleft; often incompatible with GPL combinations
- **EPL / MPL**: typically file-level or module-level obligations, with clearer boundaries than GPL

Practical takeaway:

weak copyleft can be workable in commercial stacks, but it demands rigor in how code is combined, shipped, and documented.

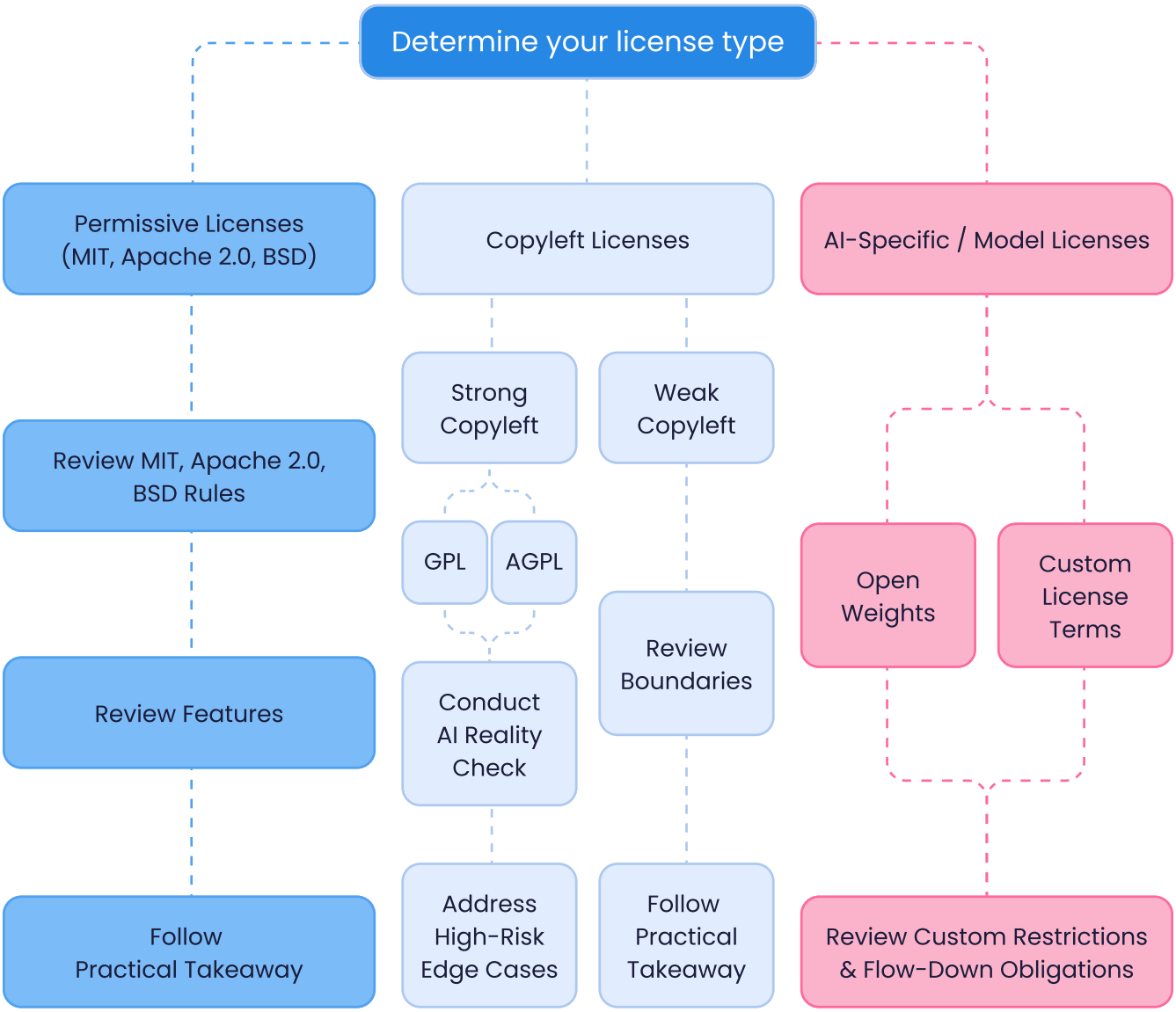
AI-Specific / Model Licenses (Open Weights and Custom Terms)

AI releases increasingly use custom licenses that may include:

- Prohibited use policies (field-of-use restrictions)
- Competitive restrictions
- Commercial thresholds (special terms for scale)
- Branding and attribution requirements
- Flow-down obligations to downstream redistributors
- Restrictions on fine-tuning, distillation, or derivative model publication

Important:

Many of these are not “open source,” even if the weights are downloadable.



04

Licensing Trends in 2026: The Rise of the "Intelligence Layer"

Permissive licenses remain the default foundation

Across modern stacks, especially cloud-native infrastructure and AI tooling, permissive licenses remain a common default because they reduce friction for adoption and productization. Apache 2.0 is frequently favored where patent clarity matters and where organizations want more explicit terms around contributions and redistribution.

Copyleft in the AI era: still strategic, more carefully scoped

Copyleft continues to play a major role in foundational projects and maintainer-led ecosystems. But businesses increasingly scrutinize copyleft in contexts where:

- The product is delivered primarily via SaaS
- The architecture involves many third-party dependencies
- Distribution boundaries are unclear (images, model packages, plugins, agents)
- The company needs to preserve proprietary differentiation (weights, orchestration, workflows)

This doesn't mean "avoid copyleft." It means teams must be intentional about where copyleft components live in the stack and how obligations are met.

Open-washing pressure and clearer definitions

As model providers market releases as “open,” procurement and legal teams increasingly separate marketing labels from licensing reality:

- Open Source AI has definitional requirements (freedoms + access to preferred form to modify).
- Many releases are better described as open weights or source-available due to missing reproducibility elements or use restrictions.

This definitional clarity is becoming a practical procurement and risk-management requirement in large organizations.

The “AI slop” contribution problem and provenance controls

Maintainers are increasingly managing high volumes of low-quality or low-traceability contributions. Common responses include:

- Stricter review standards and required tests
- Provenance signals (who authored, what tool produced it, what sources were used)
- Governance rules for AI-assisted contributions
- Security requirements (SLSA-style hardening, signed commits, stricter dependency policies)

Economic shifts: sustainability mechanisms are becoming mainstream

A growing set of initiatives aim to route value back to maintainers whose work supports commercial AI. The mechanisms vary:

- Sponsorships and foundations
- Dual licensing or paid exceptions
- Open core models
- Platform-level funding programs
- Commercial support contracts tied to critical dependencies

05

License Deep-Dives: The 2026 Perspective

The MIT License

The MIT License remains one of the most widely used permissive licenses because it is short, readable, and business-friendly.

Why it's popular

- **Simplicity first:** it's brief and easy to adopt
- **Commercially flexible:** modifications can remain proprietary
- **Standard warranty disclaimer:** "as-is" with broad liability limitations

What you must do (practical compliance)

- Include the MIT license text with copies/substantial portions
- Preserve copyright notices
- In binaries/containers, include the license in a THIRD-PARTY/NOTICES artifact that ships with the product (or is included in distribution packages)

AI context

MIT is common in high-velocity ecosystems (utilities, small libraries, research tooling). The key risk isn't the MIT terms—it's missing notices during rapid reuse, containerization, or code generation workflows.

The Apache License 2.0

Apache 2.0 is widely used in enterprise infrastructure and is often preferred in commercial environments for clearer patent language.

What makes Apache 2.0 enterprise-friendly

- **Permissive terms:** use, modify, redistribute, and sell
- **Explicit patent grant:** reduces ambiguity and improves defensibility in enterprise contexts
- **Structured compliance approach:** includes concepts like NOTICE (when provided) and clear redistribution requirements

What you must do (practical compliance)

- Include the Apache 2.0 license text in distributions
- Preserve attribution and NOTICE files where required
- Mark significant modifications where applicable (many organizations standardize this internally even when the license doesn't require per-file marking in every scenario)
- Understand patent termination concepts for inbound and outbound risk

AI context

Apache 2.0 is common in ML infrastructure, orchestration tooling, and platform components where patent defensibility matters.

Berkeley Software Distribution (BSD) 3-Clause

BSD 3-Clause is a permissive license family with an additional non-endorsement protection.

Why it's used

- Minimal restrictions and flexible redistribution
- Non-endorsement clause helps prevent downstream marketing from implying author approval

AI context

This is particularly useful when downstream products may imply endorsement of model behavior, safety, or performance.

The GNU General Public License (GPL) v3.0

GPLv3 is strong copyleft designed to keep software free and open when distribution triggers apply.

Core concept

If you distribute a derivative work of GPL-licensed code, you must provide the corresponding source under GPL terms.

Why teams call it "viral" (and why that label can mislead)

The "viral" label is shorthand for "reciprocity can extend beyond a single file," but it can obscure the real question: did you distribute a derivative work, and what exactly did you distribute?

AI context (what to say accurately in 2026)

- If GPL code is included in shipped runtime components, the compliance obligations are relatively clear.
- If GPL code appears in training pipelines or training corpora, risk analysis becomes more complex and fact-specific.
- Outputs and weights raise difficult questions about copying and derivation; treat as high-risk legal territory rather than claiming a universal rule.

Best practice

If GPL is in the stack, document boundaries, distribution models, and compliance steps early. Do not wait until release.

GNU Lesser General Public License (LGPL)

LGPL is weak copyleft designed for libraries

Core concept

- You can often link to LGPL libraries without open-sourcing your entire application
- If you modify the LGPL library itself, those modifications must be shared under LGPL terms

Dynamic vs. static linking (why it matters)

Dynamic linking is commonly preferred because it preserves the user's ability to replace/relink the LGPL component, reducing compliance complexity.

AI context

LGPL can work well for standardized math/processing libraries when you want interoperability without forcing disclosure of proprietary application logic or proprietary model artifacts.

GNU Affero General Public License (AGPL) v3.0

AGPL is the critical copyleft license for the age of cloud services and AI APIs.

Core concept

Where GPL is triggered by distribution, AGPL is designed to trigger when users interact with the software over a network (in many typical usage patterns).

AI-as-a-Service context

If you run a modified AGPL component as a web service (including an AI inference service), you may need to make the corresponding source available to users of that service.

Best practice

Treat AGPL as a deliberate product decision. If it appears in your stack, confirm whether it's in production runtime and whether the network use case creates obligations.

Common Development and Distribution License (CDDL)

CDDL is a weak copyleft license with file-level reciprocity.

Core concept

- Reciprocity applies to CDDL files and modifications to those files
- Other parts of a larger project may remain under different licenses

Compatibility note

CDDL is generally considered incompatible with GPL combinations due to structural differences in copyleft requirements.

AI context

CDDL can appear in legacy components and infrastructure-adjacent projects. The risk is usually in mixing it into GPL-controlled distributions without a careful architecture boundary.

Eclipse Public License (EPL) 2.0 (Often Overlooked, Frequently Relevant)

EPL 2.0 is a weak copyleft license commonly seen in tooling ecosystems.

Core concept

EPL often applies at a file/module boundary with specific requirements for distributing modifications.

AI context

EPL can appear in developer tools and integration frameworks. It is rarely a "blocker," but it demands correct attribution, notice handling, and boundary clarity when distributing modified components.

Mozilla Public License (MPL) 2.0 (File-Level Copyleft with Clear Boundaries)

MPL is designed to be business-friendly while preserving openness at the file level.

Core concept

- Changes to MPL-covered files must be shared under MPL
- You can combine MPL code with proprietary code as long as boundaries remain clear

AI context

MPL can be a strong compromise when you want an ecosystem to remain open without forcing full-stack disclosure.

Open Weights and Model Licenses (AI-Specific)

Open weights and custom model licenses are increasingly common. They can be extremely useful—but they are not equivalent to OSI open source.

Common patterns

- Use restrictions: prohibited use policies and field-of-use restrictions
- Commercial thresholds: special terms or separate agreements for high-scale usage
- Redistribution rules: restrictions on sharing weights or derivatives
- Fine-tuning and distillation terms: restrictions on publishing modified models
- Flow-down obligations: downstream users must accept the same terms.

Weight transparency vs. reproducibility

Weights alone do not necessarily provide enough information to reproduce the system or validate training provenance. For many organizations, this drives additional requirements: model cards, data lineage, evaluation documentation, and internal risk approvals

Practical compliance checklist (models)

- Track model licenses as first-class artifacts (not “just another dependency”)
- Maintain a model inventory: name, version, license, usage scope, distribution method, and downstream obligations
- Record fine-tuning sources and constraints
- Ensure prohibited-use obligations do not conflict with your product use cases
- Ensure redistribution rules match your delivery model (on-prem, edge, customer export, API)

06 The 2026 License Cheat Sheet

License	Type	Must Distribute Source?	Patent Grant?	Notable 2026 AI Context
MIT	Permissive	No	Not Explicit*	Common in AI tooling; biggest risk is missing notices in fast-moving pipelines
APACHE 2.0	Permissive	No	Yes	Often preferred in enterprise AI due to explicit patent terms and structured notice handling.
BSD 3 - CLAUSE	Permissive	No	No**	Non-endorsement helps prevent downstream marketing implying author approval of model behavior.
GPL V3.0	Strong Copyleft	Yes (when triggered)	Yes	Clear obligations when distributed in runtime; training/ weights/outputs are risk-heavy and fact-specific.
LGPL	Weak Copyleft	Yes (for modified library)	Yes***	Useful for library boundaries; dynamic linking often reduces compliance complexity.
AGPL V3.0	Strong copyleft	Yes	Yes	High impact for SaaS/API delivery (including inference services).
CDDL	Weak copyleft	Yes (file level)	Yes	File-level reciprocity; mixing with GPL is commonly considered incompatible.
EPL 2.0	Weak copyleft	Yes	Yes	Often appears in tooling ecosystems; manage boundaries and notice obligations.
MPL 2.0	Weak copyleft	Yes	Yes	File-level copyleft with clearer commercial boundaries than GPL-style linking debates.
OPEN WEIGHTS / MODEL LICENSES	Mixed / Custom	Partial / Varies	Varies	Often includes use restrictions, redistribution rules, and commercial thresholds; not necessarily OSI open.

*Note: Some versions of MIT include patent grants, but the standard version does not.

**Note: Some permissive licenses include patent language, but BSD 3-Clause does not explicitly grant patents in its classic form.

***Note: LGPL inherits patent concepts via the GPL family context and terms; specifics depend on version and combination.

07

2026 "Red Flags" & Next Steps

Red Flag 1: Missing notices and attribution

One of the most common and preventable compliance failures is incomplete third-party notices—missing license texts, missing attribution lines, missing NOTICE files, or missing disclaimers—especially when code is copied across repos, containers, build pipelines, and AI-assisted development workflows.

What to do

- Standardize THIRD-PARTY/NOTICES generation for every shipping artifact (apps, containers, installers, on-prem bundles)
- Make "notice hygiene" an automated build requirement, not a manual checklist at release time

Red Flag 2: Confusing "open weights" with open source

If your procurement process treats "downloadable weights" as "open source," you may inherit hidden obligations or restrictions (use limits, redistribution constraints, competitive restrictions, or incompatible use policies).

What to do

- Categorize every AI artifact as: Open Source AI, open weights, or source-available/custom
- Require legal review for any model license with use restrictions or commercial thresholds
- Confirm redistribution rights align with your product delivery model

Red Flag 3: Copyleft surprises in SaaS and model serving

AGPL, GPL, and weak copyleft licenses can create obligations that surprise teams—especially when a model server, inference gateway, or orchestration layer is offered via API.

What to do

- Identify copyleft components in runtime and model-serving paths early
- Confirm whether network delivery changes obligations (AGPL risk)
- Architect boundaries deliberately, and document them

Red Flag 4: Regulatory documentation pressure is increasing

AI regulation is pushing organizations toward stronger documentation, lifecycle controls, and risk management—especially for high-risk systems.

What to do

- Treat compliance as an engineering system: policies, inventories, documentation templates, testing, monitoring, and audit trails
- Align internal documentation with what regulators and enterprise customers ask for (technical documentation, intended use, known limitations, risk controls, incident reporting procedures)

Next steps: A practical compliance workflow

1. Inventory everything: code dependencies + model artifacts + datasets + deployment methods
2. Classify risk: permissive vs copyleft vs custom model terms; distribution vs SaaS triggers; prohibited-use conflicts
3. Automate SBOM + Model BOM: treat models like dependencies with their own license and obligations
4. Enforce policy in CI/CD: block prohibited licenses; require notices; require approvals for restricted model licenses
5. Provenance and governance: record AI-assisted contribution signals where your org requires it; use signed commits and review gates
6. Release discipline: ship notices with every artifact; verify redistribution rights before export or on-prem delivery

Mend.io is a leading application security solution that helps organizations fix less and reduce risk faster. Built for both AI-driven and modern development workflows, **Mend.io** gives teams visibility into all code – human-written, AI-generated, open source, third-party and container components – and helps them prioritize and remediate the risks that matter most.

Visit our website