AETHER

Open Source (Objective A)

High-performance memory-safe web browser

Contents

Introduction 2

Part 1: Legal framework and preparation for the opening 2

1.1 Choice and justification of the license (LICENSE) 2

1.2 Axis chosen: memory consumption (RAM) 2

1.3 Onboarding and reception documentation 3

Part 2: Breakdown and structuring of the initial issues 3

2.1 Idk 3

2.2 Idk 3

Part 3: Sourcing and calls for contribution's campaigns 4

3.1 Targeted channels and communities 4

3.2 Outreach Register 4

Part 4: Management, review and processing of contributions 4

4.1 Monitoring procedure and SLA 4

4.2 Accessibility metrics and testing 4

4.3 Recognition of contributors 4

Part 5: Schedule and deliverables (Planning) 4

Introduction

This document covers Complementary Objective A of the Technical Track (Tech4): Opening the Aether project to external contributions. It defines the strategy, governance guidelines, tools, and community engagement timeline for making the Aether browser accessible and usable by external developers. In accordance with the Student Guide, this objective complements the existing community contribution mechanisms of Objective 1, focusing on creating an open and sustainable ecosystem around the official Aether repository. Status: Initial version. The licensing strategy, onboarding documentation, and issue breakdown are established for completion by the end of October 2026. Calls for contributions and proposal processing will continue until May 2027 at least.

Part 1: Legal framework and preparation for the opening

1.1 Choice and justification of the license (LICENSE)

The choice of software license governs the rights granted to contributors and reusers of Aether's code.

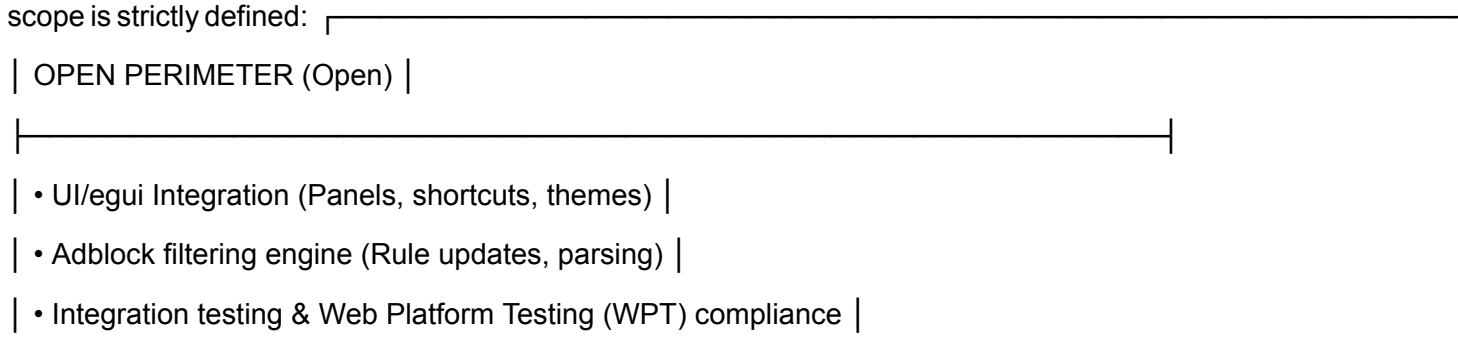
Selected license: Mozilla Public License 2.0 (MPL 2.0).

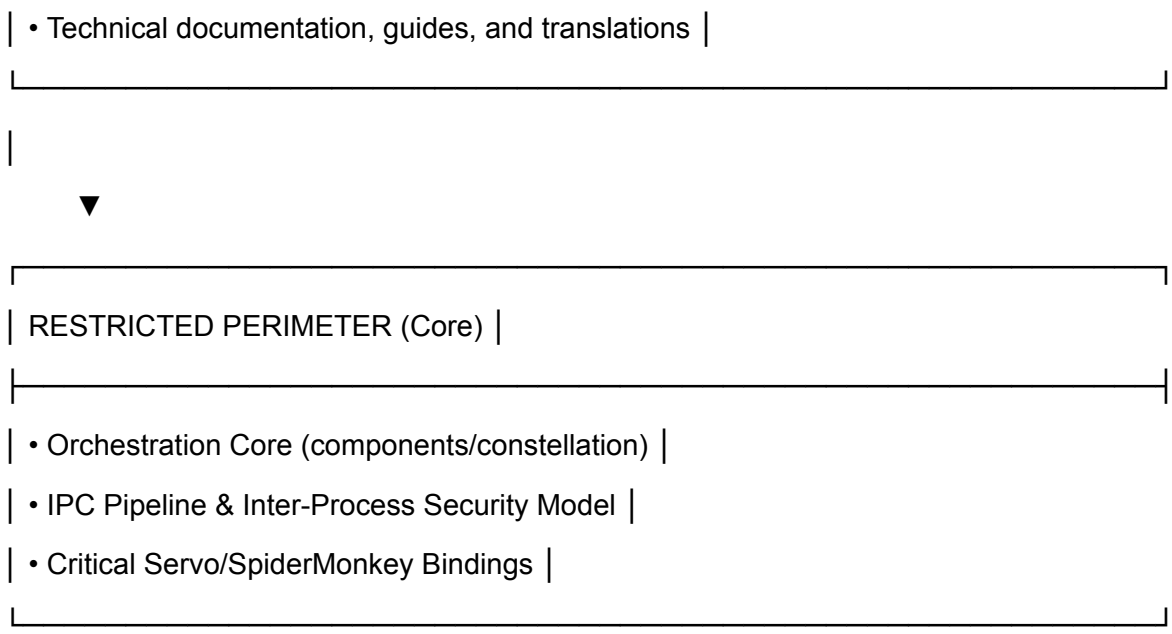
Impact analysis and permissions:

- What it allows: Anyone can freely copy, distribute, modify, and integrate Aether's code into open-source or proprietary projects.
- What it requires: Any existing source file (or direct modification of an existing file under MPL) must remain open and redistributed under the same MPL 2.0 license. New, separate proprietary files or modules, however, can be distributed under other licenses (low-reciprocity license / file-level copyleft).
- Strategic justification: This choice guarantees perfect alignment and direct compatibility with the Servo rendering engine (itself under MPL 2.0), which forms the foundation of Aether. It protects contributions made to Aether's core components while offering optimal flexibility for integrating external dependencies from the Rust ecosystem.

1.2 Axis chosen: memory consumption (RAM)

To preserve the browser's architectural consistency while encouraging external participation, the scope is strictly defined:





Open to contributions: Interface modules (aether/ports), Adblock engine (components/net/adblock), WPT tests (tests/wpt) and documentation. Under strict team control: The overall orchestration architecture (components/constellation) and the inter-process memory model.

1.3 Onboarding and reception documentation

Three major files are maintained in the repository root:

1. README.md (Contributor): Project overview, system requirements, local build instructions (mach build), and test execution instructions (mach test).
2. CONTRIBUTING.md:
    - Rust coding conventions (mach fmt).
    - Commit message format (standard Conventional Commits).
    - Pull Request (PR) submission procedure and acceptance criteria defined in the task in the Github Project.
3. CODE_OF_CONDUCT.md: Rules of conduct within the Aether community.

Part 2: Breakdown and structuring of the initial issues

2.1 Idk

TODO!

2.2 Idk

TODO!

Part 3: Sourcing and calls for contribution's campaigns

3.1 Targeted channels and communities

TODO!

3.2 Outreach Register

TODO!

Part 4: Management, review and processing of contributions

4.1 Monitoring procedure and SLA

All contributions received are rigorously monitored to ensure a seamless experience:

- Time to First Response (TFR): Measured and recorded in each PR.
- Strict review policy: No PR is merged blindly. At least one reasoned code review exchange is required.
- Justified rejections or modifications: In case of non-compliance with performance or security requirements, a refactoring is requested or a reasoned rejection is publicly and respectfully provided.

4.2 Accessibility metrics and testing

To measure the effectiveness of the approach, two main indicators are tracked:

1. Repository metrics: Number of views, clones, stars, and unique contributors.
2. External accessibility test: Protocol:
    - A person outside the team attempts to install the project and resolve an issue using only the documentation (README.md and CONTRIBUTING.md).
    - Test deliverables: A dated report specifying the installation time, identified roadblocks, and documentation fixes.

4.3 Recognition of contributors

Contributors who have submitted accepted code or documentation are publicly credited:

- A dedicated ”Contributors” section on GitHub.
- Explicit mention in the release notes.

Part 5: Schedule and deliverables (Planning)

Document Date: September 29, 2026

Modify Date: November 13, 2026