# AEP Versioning and Stability Proposal

Author: yusuke.tsutsumi@gm.com +yourname
Contributors: Alex Stephen rfrankel@roblox.com Dan.Hudlow, Mike.Kistler, Marsh.Gardiner
Created: 2025-05-27
Updated: 2025-07-25

# Summary

This document proposes the guarantees around versioning and stability for the https://aep.dev/ project.

# Proposal

This document proposes the following:

## AEPs will use an edition-based versioning scheme

AEPs will be published in editions, similar (but not identical) to specifications such as for the C++ or Rust programming language:

- Changes to the specification will only be made when the next version is published.
- The naming of the version will be based on the year in which it is published (e.g. aep-2025).
- Upon publishing the current edition, the next preview edition will be created (e.g. aep-2027-preview). This preview edition will accumulate changes until it is published as the next edition.
- AEP editions may have patch version updates which adhere to semantic versioning (e.g. aep-2025.25). The patch version will only contain fixes to typos and clarification on existing guidance.
- Editions will be published every 2 years.

Although not strictly related to versioning, the AEP project will generally strive to minimize breaking changes, even across editions.

## AEP first-party client and tool guidance

This proposal only applies to AEP first-party clients. Third party open source projects or organizations are governed outside of this project, and may have their own guarantees.

### Clients will follow semantic versioning

Clients will adhere to Semantic Version 2.0.

### The newest major version of clients and tools will be compatible with, at minimum, the 3 latest AEP editions

The following guidance applies to the most recent major versions of clients:

- Clients may provide different support guarantees for older major versions.
- Each major version will state what AEP editions are supported by those clients.
- This guidance does not apply to accepting PRs on older major versions: those may be accepted for any maintained branch of the project, at the discretion of the client maintainers.
- Clients and tools may support features in preview editions, but support for preview edition features in all clients and tools are not guaranteed.

# Goals

In order to meet the needs of enterprises that wish to adopt AEPs, we must provide stability guarantees and versioning. These help enterprises understand the rate of change, and therefore level of effort for maintenance, to adhere to the specification, as well as how long they can expect to leverage the ecosystem of the project when adopting a major version.

Because AEPs provide a discipline around designing and implementing interfaces, potentially coordinated across many different teams, it is existential for the project to clearly define:

1. The constraints that discourage disruptive changes; and
2. The mechanisms to communicate those changes as the project evolves.

# Design Details

## Why an edition-based scheme?

The AEPs have two goals that are difficult to achieve in concert:

1. Providing a set of modern best practices for remote APIs.
2. Providing a stable ecosystem of tooling that organizations can adopt for their use.

This is due to the need to constantly evolve the best practices, which may contradict older best practices and therefore result in a breaking change. These breaking changes can be difficult for both services producing these APIs to adopt, as well as complicate clients with multiple different code paths to handle these different versions of clients.

An edition-based system will help provide clear expectations around the cadence in which breaking changes **could** be introduced, as well as act as an anchor on which other durations could be based (for example, support for a number of editions in major versions of clients).

## Why are clients versioned separately?

Although clients are expected to support recent AEP editions and could have a similar versioning scheme, clients may also need to introduce breaking changes for a variety of reasons unrelated to a new AEP edition, including:

1. An interface change in the client itself.
2. A change to support a new integration or interface (for example, supporting a new version of the Terraform SDK, or a major version of the MCP server protocol).

This necessitates the ability to express these changes to consumers. As such, decoupling the client version from the AEP editions is a critical requirement.

## AEP Client Prioritization

The following does not serve as a guarantee, but outlines a loose prioritization that was used to inform the versioning guarantees

1. Security updates (for all supported versions).
2. Support for new features in the specification.
3. If a client is not deprecated or EOL, it will support the current version of the spec.
4. Backporting features to older versions of the spec.

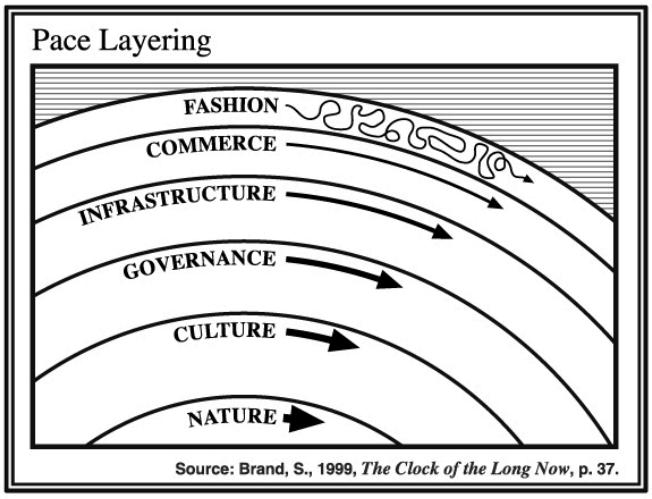
## Examples of API specification changes

Although it has been decided that *all* changes to the specification (sans clarification or typos) will be reserved until the next edition, the following table enumerates some examples of API specification changes.

| Change Description | Would it have to wait until the next edition? |
|---|---|
| renaming a field (e.g. name to path) | yes |
| change the syntax for a filter or query language | yes |
| updating versioning guidance | yes |
| adding a new standard method | yes: someone could have been using PUT (apply) before, and now they break the guidance. |
| adding a new field to an aep-owned proto/OAS extension | no |

| Change Description | Would it have to wait until the next edition? |
|---|---|
| removing guidance | yes: although the specification can remove guidance without breaking the adherence of the API, a client may rely on that guidance, and a change to remove guidance may cause a client to break (in that it can no longer rely on that guidance). |
| updating a design pattern (e.g. singletons or revisions) | yes |

# Appendix




# Scratch

## User Journeys

- As an organization, we would like to adopt a specific AEP edition, and are worried about how long we can expect the first-party ecosystem will support this AEP edition.
    - They want the CLI, UI, MCP server, and Terraform Provider, and linter.
    - What are they worried about?
        - There is a security vulnerability in a client.
        - An SDK / integration (e.g. MCP or Terraform) requires updates, and the AEP first-party client for that edition does not provide that integrations

        - There's a shiny new thing I want to use, but the AEP first-party client for that edition doesn't support it

- As an organization, we would like to adopt AEPs, and are worried the whole project won't gain critical mass and the promise of the interface standardization and shared tooling will fade away.
- As an organization, we would like to adopt a specific AEP edition, but we have to extend it and fork the tools because it's too naive to handle our whole problem space, and I'm worried that it will be a ton of work to port my extensions to the tools that support the next AEP edition.
- As an organization, we would like to adopt a specific AEP edition, and we are worried that we and our customers will develop deep enough dependencies on the APIs conforming to this edition, that we'll never be able to update and will eventually be forced to support deprecated versions of all the tooling ourselves.

| | AEP Edition 2025 | AEP Edition 2029 |
|---|---|---|
| Client Version 2.x | No | Yes |
| Client Version 1.x | Yes | No |

What are our priorities around the 1.x branch?

- security updates only for 4 years.
- we will still accept patches for older versions.

# 2025-06-13 with Richard and Mike

- the AEP edition will be supported for N years - let's define that
- Richard: if we publish a new version of the client, our uses may not want / be able to update their client.
    - client support guarantees are also important.
- Mike:
    - producer workflows will have to support all editions, to not break user workflows to produce API.
    - at MSFT: if an azure API goes GA, it's supported indefinitely.
    - 10 years security updates minimum for a major version of a client from the point of publication.

    - An AEP edition will be supported by the most recent major versions of tools for N years.
- Rich:
    - Three separate/orthogonal numbers:
        - Security update support window (10 years, above)
        - How far back do major client versions (e.g. aepc, aep-cli) go in terms of supporting older editions of the spec?  The most recent N specs (N probably 1 or 2)?
        - What's the *minimum* time between AEP editions?  (2 years, 4 years?)
- edition naming: 2025 -> 2029-preview -> 2029