



Network ▾

Nodes ▾

APIs ▾

Tools ▾

ACPs ▾

☐ What is an Avalanche Community Proposal (ACP)? ▾

Avalanche Community Proposals (ACPs)

Official framework for proposing improvements and gathering consensus around changes to the Avalanche Network



What is an Avalanche Community Proposal (ACP)?

An Avalanche Community Proposal is a concise document that introduces a change or best practice for adoption on the Avalanche Network. ACPs should provide clear technical specifications of any proposals and a compelling rationale for their adoption.

ACPs are an open framework for proposing improvements and gathering consensus around changes to the Avalanche Network. ACPs can be proposed by anyone and will be merged into this repository as long as they are well-formatted and coherent. Once an overwhelming majority of the Avalanche Network/Community have signaled their support for an ACP, it may be scheduled for activation on the Avalanche Network by Avalanche Network Clients (ANCs). It is ultimately up to members of the Avalanche Network/Community to adopt ACPs they support by running a compatible ANC, such as AvalancheGo.



Network ▾

Nodes ▾

APIs ▾

Tools ▾

ACPs ▾

- A **Standards Track** ACP describes a change to the design or function of the Avalanche Network, such as a change to the P2P networking protocol, P-Chain design, Subnet architecture, or any change/addition that affects the interoperability of Avalanche Network Clients (ANCs).
- A **Best Practices Track** ACP describes a design pattern or common interface that should be used across the Avalanche Network to make it easier to integrate with Avalanche or for Subnets to interoperate with each other. This would include things like proposing a smart contract interface, not proposing a change to how smart contracts are executed.
- A **Meta Track** ACP describes a change to the ACP process or suggests a new way for the Avalanche Community to collaborate.
- A **Subnet Track** ACP describes a change to a particular Subnet. This would include things like configuration changes or coordinated Subnet upgrades.

ACP Statuses

There are four statuses of an ACP:

- A **Proposed** ACP has been merged into the main branch of the ACP repository. It is actively being discussed by the Avalanche Community and may be modified based on feedback.
- An **Implementable** ACP is considered "ready for implementation" by the author(s) and will no longer change meaningfully from its current form (which would require a new ACP).
- An **Activated** ACP has been activated on the Avalanche Network via a coordinated upgrade by the Avalanche Community. Once an ACP is **Activated**, it is locked.
- A **Stale** ACP has been abandoned by its author(s) because it is not supported by the Avalanche Community or has been replaced with another ACP.



[Network](#) ▾[Nodes](#) ▾[APIs](#) ▾[Tools](#) ▾[ACPs](#) ▾

The ACP process begins with a new idea for Avalanche. Each potential ACP must have an author(s): someone who writes the ACP using the style and format described below, shepherds the associated GitHub Discussion, and attempts to build consensus around the idea. Note that ideas and any resulting ACP is public. Authors should not post any ideas or anything in an ACP that the Author wants to keep confidential or to keep ownership rights in (such as intellectual property rights).

Step 1: Post Your Idea to GitHub Discussions

The author(s) should first attempt to ascertain whether there is support for their idea by posting in the "Ideas" category of GitHub Discussions. Vetting an idea publicly before going as far as writing an ACP is meant to save both the potential author(s) and the wider Avalanche Community time. Asking the Avalanche Community first if an idea is original helps prevent too much time being spent on something that is guaranteed to be rejected based on prior discussions (searching the Internet does not always do the trick). It also helps to make sure the idea is applicable to the entire community and not just the author(s). Small enhancements or patches often don't need standardization between multiple projects; these don't need an ACP and should be injected into the relevant development workflow with a patch submission to the applicable ANC issue tracker.

Step 2: Propose an ACP via Pull Request

Once the author(s) feels confident that an idea has a decent chance of acceptance, an ACP should be drafted and submitted as a pull request (PR). This draft must be written in ACP style as described below. It is highly recommended that a single ACP contain a single key proposal or new idea. The more focused the ACP, the more successful it tends to be. If in doubt, split your ACP into several well-focused ones. The PR number of the ACP will become its assigned number.

Step 3: Build Consensus on GitHub Discussions and Provide an Implementation (if Applicable)



[Network](#) ▾[Nodes](#) ▾[APIs](#) ▾[Tools](#) ▾[ACPs](#) ▾

discussion. It is recommended for author(s) or supportive Avalanche Community members to post an accompanying non-technical overview of their ACP for general consumption in this GitHub Discussion. The ACP should be reviewed and broadly supported before a reference implementation is started, again to avoid wasting the author(s) and the Avalanche Community's time, unless a reference implementation will aid people in studying the ACP.

Step 4: Mark ACP as `Implementable` via Pull Request

Once an ACP is considered complete by the author(s), it should be marked as `Implementable`. At this point, all open questions should be addressed and an associated reference implementation should be provided (if applicable). As mentioned earlier, the Avalanche Foundation meets periodically to recommend the ratification of specific ACPs but it is ultimately up to members of the Avalanche Network/Community to adopt ACPs they support by running a compatible Avalanche Network Client (ANC), such as AvalancheGo.

[Optional] Step 5: Mark ACP as `Stale` via Pull Request

An ACP can be superseded by a different ACP, rendering the original obsolete. If this occurs, the original ACP will be marked as `Stale`. ACPs may also be marked as `Stale` if the author(s) abandon work on it for a prolonged period of time (12+ months). ACPs may be reopened and moved back to `Proposed` if the author(s) restart work.

Maintenance

ACP maintainers will only merge PRs updating an ACP if it is created or approved by at least one of the author(s). ACP maintainers are not responsible for ensuring ACP author(s) approve the PR. ACP author(s) are expected to review PRs that target their unlocked ACP (`Proposed` or `Implementable`). Any PRs opened against a locked ACP (`Activated` or `Stale`) will not be merged by ACP maintainers.



[Network](#) ▾[Nodes](#) ▾[APIs](#) ▾[Tools](#) ▾[ACPs](#) ▾

- **Preamble**: Markdown table containing metadata about the ACP, including the ACP number, a short descriptive title, the author(s), and optionally the contact info for each author, etc.
- **Abstract**: Concise (~200 word) description of the ACP
- **Motivation**: Rationale for adopting the ACP and the specific issue/challenge/opportunity it addresses
- **Specification**: Complete description of the semantics of any change should allow any ANC/Avalanche Community member to implement the ACP
- **Security Considerations**: Security implications of the proposed ACP

Each ACP can have the following parts:

- **Open Questions**: Questions that should be resolved before implementation

Each **Standards Track** ACP must have the following parts:

- **Backwards Compatibility**: List of backwards incompatible changes required to implement the ACP and their impact on the Avalanche Community
- **Reference Implementation**: Code, documentation, and telemetry (from a local network) of the ACP change

Each **Best Practices Track** ACP can have the following parts:

- **Backwards Compatibility**: List of backwards incompatible changes required to implement the ACP and their impact on the Avalanche Community
- **Reference Implementation**: Code, documentation, and telemetry (from a local network) of the ACP change



ACP Formats and Templates

Network ▾

Nodes ▾

APIs ▾

Tools ▾

ACPs ▾

Auxiliary Files

ACPs may include auxiliary files such as diagrams or code snippets. Such files should be stored in the ACP's subdirectory (`ACPs/N-T/*`). There is no required naming convention for auxiliary files.

Waived Copyright

ACP authors must waive any copyright claims before an ACP will be merged into the repository. This can be done by including the following text in an ACP:

```
## Copyright
```



```
Copyright and related rights waived via [CC0](https://creativecommons.org/publicdomain/zero/1.0/)
```

Proposals

You can view the status of each ACP on the [ACP Tracker](#).

Number	Title	Author(s)	Type
<u>13</u>	Subnet-Only Validators (SOVs)	Patrick O'Grady (contact@patrickogrady.xyz)	Standards
<u>20</u>	Ed25519 p2p	Dhruba Basu (@dhrubabasu)	Standards
<u>23</u>	P-Chain Native Transfers	Dhruba Basu (@dhrubabasu)	Standards
<u>24</u>	Activate Shanghai EIPs on C-Chain	Darioush Jalali (@darioush)	Standards



Network ▾

Nodes ▾

APIs ▾

Tools ▾

ACPs ▾

<u>30</u>	Integrate Avalanche Warp Messaging into the EVM	Aaron Buchwald (aaron.buchwald56@gmail.com)	Standards
<u>31</u>	Enable Subnet Ownership Transfer	Dhruba Basu (@dhrubabasu)	Standards
<u>41</u>	Remove Pending Stakers	Dhruba Basu (@dhrubabasu)	Standards
<u>62</u>	Disable <code>AddValidatorTx</code> and <code>AddDelegatorTx</code>	Jacob Everly (https://twitter.com/JacobEv3rly), Dhruba Basu (@dhrubabasu)	Standards
<u>75</u>	Acceptance Proofs	Joshua Kim (@joshua-kim)	Standards
<u>77</u>	Reinventing Subnets	Dhruba Basu (@dhrubabasu)	Standards
<u>83</u>	Dynamic Multidimensional Fees for P-Chain and X-Chain	Alberto Benegiamo (@abi87)	Standards
<u>84</u>	Table Preamble for ACPs	Gauthier Leonard (@Nuttymoon)	Meta
<u>99</u>	Validator Manager Solidity Standard	Gauthier Leonard (@Nuttymoon), Cam Schultz (@cam-schultz)	Best Practices
<u>103</u>	Add Dynamic Fees to the X-Chain and P-Chain	Dhruba Basu (@dhrubabasu), Alberto Benegiamo (@abi87), Stephen Buttolph (@StephenButtolph)	Standards
<u>108</u>	EVM Event Importing	Michael Kaplan (@michaelkaplan13)	Best Practices
<u>113</u>	Provable Virtual Machine Randomness	Tsachi Herman (@tsachiherman)	Standards
<u>118</u>	Standardized P2P Warp Signature Request Interface	Cam Schultz (@cam-schultz)	Best Practices

Network ▾

Nodes ▾

APIs ▾

Tools ▾

ACPs ▾

	1 nAVAX	Ceyhan Onur (@ceyonur)	
<u>131</u>	Activate Cancun EIPs on C-Chain and Subnet-EVM chains	Darioush Jalali (@darioush), Ceyhan Onur (@ceyonur)	Standards
<u>151</u>	Use current block P-Chain height as context for state verification	Ian Suvak (@iansuvak)	Standards
<u>176</u>	Dynamic EVM Gas Limits and Price Discovery Updates	Stephen Buttolph (@StephenButtolph), Michael Kaplan (@michaelkaplan13)	Standards
<u>181</u>	P-Chain Epoched Views	Cam Schultz (@cam-schultz)	Standards
<u>191</u>	Seamless L1 Creations (CreateL1Tx)	Martin Eckardt (@martineckardt), Aaron Buchwald (aaronbuchwald), Michael Kaplan (@michaelkaplan13), Meag FitzGerald (@meaghanfitzgerald)	Standards
<u>194</u>	Continuous Execution	Arran Schlosberg (@ARR4N), Stephen Buttolph (@StephenButtolph)	Standards
<u>204</u>	Precompile for secp256r1 Curve Support	Santiago Cammi (@scammi), Arran Schlosberg (@ARR4N)	Standards
<u>209</u>	EIP-7702-style Set Code for EOAs	Stephen Buttolph (@StephenButtolph), Arran Schlosberg (@ARR4N), Aaron Buchwald (aaronbuchwald), Michael Kaplan (@michaelkaplan13)	Standards
<u>224</u>	Introduce ACP-176-Based Dynamic Gas Limits and Fee Manager Precompile in Subnet-EVM	Ceyhan Onur (@ceyonur), Michael Kaplan (@michaelkaplan13)	Standards



Network ▾

Nodes ▾

APIs ▾

Tools ▾

ACPs ▾

<u>236</u>	Auto-Renewed Staking	Razvan Angheluta (@rrazvan1)	Standards
<u>247</u>	Delegation Multiplier Increase & Maximum Validator Weight Reduction	Giacomo Barbieri (@ijaack94), BENQI (@benqifinance)	Standards
<u>256</u>	Update Hardware Requirements for Primary Network Nodes	Aaron Buchwald (@aaronbuchwald), Martin Eckardt (@martineckardt), Meaghan FitzGerald (@meaghanfitzgerald)	Best Practices
<u>267</u>	Increase Validator Uptime Requirement from 80% to 90%	Martin Eckardt (@martineckardt)	Best Practices
<u>273</u>	Reduce Minimum Validator	Eric Lu (@ericlu-avax), Martin Eckardt (@martineckardt), Meaghan FitzGerald	Standards

Is this guide helpful?

 Yes No Copy Markdown

ACP-285: Reduce Minimum Consumption Rate >

Details for Avalanche Community Proposal 285: Reduce Minimum Consumption Rate

