HIP-12 Hacash development workflow an...

Due to Hacash's completely decentralized goal and status, there is no pre-mining, no ICO, no development tax, no capital investment, and no foundation to operate, the development work is not funded in any way, and for a long time (almost four years) the development team is completely in a state of free and voluntary code contribution to the community. The development work fully follows the vision of the Hacash white paper and is in line with the community consensus to make the mechanism fair and secure. All source code is 100% open source and is assembled in a repository at https://github.com/hacash.

From Hacash's Github commit log, as of February 2023, Jojoin (Hacash's main core developer) has contributed over 90% of the code, including the mainnet, wallet and SDK, and development documentation. Due to the slow development of Hacash, it has not gained the attention of the mainstream market yet, and no more people are involved in the development of Hacash, nor has it formed a developer community with scale. The core code of the main network is currently managed by Jojoin under one account.

The goal of this proposal is to form a complete Hacash development workflow by referring to the source code management mechanisms of mature crypto projects such as Bitcoin and Ethereum, as well as to define the classification of Hacash development team members, their participation qualifications, and manage permissions.

I. Development team structure

Hacash's development team is organized into three tiers.

1. Core Group
2. Core Prep Group
3. Application Group

There is no limit to the number of Application Group and the number of participants per group. Application groups can form multiple level groups around different sections or topics, such as wallet groups, block explorer groups, channel chain wallet groups, etc. When the number of participants in a single application group is too large, or the development workload is too large, which affects the communication and cooperation between each other, it can be split into two or

more application groups, and a developer can also participate in multiple application groups at the same time. The leader of each application group is a member of the Core Group or Core Prep Group.

The Core Prep Group has no more than 25 members. The main tasks are to review the code submitted by each Application Group member as the group leader, check for potential security vulnerabilities or performance issues, participate in the design and discussion of the public chain core and payment and scaling protocol layers, and submit experimental or unit test code for the public chain layer, as well as pre-review the code submitted to the public chain layer.

The Core Group, with no more than 5 members, has the authority to merge the main version of the public chain layer code, and has direct decision-making power on each architectural scheme and protocol design. It has direct merging authority for the main network code submission that does not change the core value consensus, and voting authority for major matters such as incompatible upgrades of public chains, protocol underlay modifications and monetary value model changes. The proportion of valid votes for major upgrades involving the monetary value layer and core protocol layer of public chains, and changes that can be approved is: 1/1, 2/2, 2/3, 3/4 or 3/5. Abstentions or overdue votes are considered negative votes.

The above hierarchy does not represent a distinction in the amount of code contributed, but rather a distinction in source code merging authority. It is possible to be a very important code contributor without being a member of the core group, except that do not have code review and source code management responsibilities.

To become a member of each level of the group, a number of conditions need to be met:

Application Group:

No restrictions. The merge code authority is managed by the group leader and deputy group leader.

Core Prep Group:

1. Proficiency in one of the programming languages C++, Java, Golang or Rust.
2. Contribute 6000 lines of source code to the Hacash application layer, or 2000 lines of source code or more to the Hacash public chain layer
3. Independently developed a 100 star or more pure source code open source projects (non-data collation or documentation class)

4. Familiar with the technical route, monetary theory and ultimate goal of Hacash, and preliminary understanding of the underlying architecture of Hacash public chain

Core group:

1. Proficient in at least two of C++, Golang or Rust programming languages
2. Have contributed more than 10,000 lines of code to the Hacash public chain layer and have made major refactorings or upgrades to major modules on the underlying public chain. Have sufficient time (no less than 20 hours per week) to be responsible for technical management and key decisions of Hacash
3. Developed and managed 2 or more open source projects with 200 star, Have contributed code to 5 or more open source projects
4. Fully familiar with the underlying infrastructure and implementation details of Hacash public chain, and have the ability to design and refactor at the architectural level, and have participated in the design and development of communication protocols, payment interfaces or p2p network protocols.
5. Fully study all the theories and technical materials of Hacash, master the underlying essence and core value source of Hacash, fully understand the decentralization and fairness route, technical direction and monetary theory of Hacash, understand the trade-offs, limitations and boundaries of Hacash in all aspects, and agree with the ultimate mission of Hacash
6. Accumulate a certain level of social credibility throughout the Hacash community, demonstrating excellent character and adequate work ethic. Gain the trust of the community by maintaining complete fairness and integrity at all times, by not representing any unilateral position, by not seeking unfair advantage for any individual or group, by not doing anything that would directly or potentially harm the long-term development of the project, and by not seeking to overturn the consensus of values that Hacash has gained for personal gain

II . Development workflow

Hacash's development work is divided into four categories, each corresponding to a different authority and process.

1. Application layer development
2. Public chain layer bug fix
3. Public chain layer new feature development
4. Public chain core value layer or protocol layer change

For the first type of development work, Final review and code merging will be done by the application team leader. The core prep and core groups maintain the authority to override and roll back code submitted by the application groups.

The second type of development work is done by the core preparation group to complete the code review, and the code merging is done directly by the core group members.

For the third type of development work, the core preparation group performs the first code review and submits it to the core group to complete the final code review, and then the core group members perform the code merge after a brief discussion and communication.

The fourth category of development work, which requires the participation of the entire Hacash community and is performed in conjunction with Hacash's HIP system (Hacash Improvement Proposal), follows the following process.

1. A new HIP is proposed by the community or development group, and the topic is discussed on Github, Discord, the HacashTalk forums, or other public communities of Hacash, to get feedback from the whole community.
2. The person who proposed this HIP is responsible for summarizing and organizing all the discussion materials into a preliminary reference document.
3. Continue to discuss the details of various aspects of the proposal, provide test or validation code, and develop a more complete draft HIP after no major controversies or gaps are apparent.
4. Make the draft HIP available to all channels that can reach the Hacash community, and after a period of time, with no more objections from the community, submit it to the core group members for a vote.
5. If the core development team votes to approve, the HIP will be announced to the community as approved and development work will be scheduled accordingly.

The initiator is responsible for the promotion of the above HIP process, the collation of each discussion and material, and the release of documents and drafts.