ZEP Token Mechanics

Alejo Salles
alejo@zeppelinos.org
https://zeppelinos.org

October 2018
WORKING DRAFT

ZeppelinOS a platform to develop, deploy and operate smart contract projects on Ethereum and every other EVM and eWASM-powered blockchain. Its native token, ZEP, aligns incentives where required to create a healthy ecosystem of secure smart contract projects. Here, we present a summary of ZeppelinOS's token mechanics by first discussing ZEP's central role in the curated EVM package system, and then describing its more direct usage in other parts of the platform.

1 The Package System

ZeppelinOS provides a set of high-quality EVM packages covering a vast range of functionality like token and other EIP standards, math libraries, data structures, oracles, wallets, voting, token curated registries, or state channels, to name a few. All of these can be seamlessly linked by developers to their own projects. Package developers, in turn, can find in ZeppelinOS a platform for monetizing their projects. When publishing their packages to the platform, they must back them by locking towards them an amount of ZEP tokens (vouching), thus signaling their support for new contributions. They can also choose to charge a linking fee, which users will need to pay if they want to use the packages through the platform. All actions available in the platform are mediated by ZEP, and are described in more detail below.

1.1 Publishing New Packages

Developers vouch ZEP beyond a minimum spam-preventing amount to publish a new package, and choose an optional linking fee. This allows them to set up their packages as microbusinesses, profiting from ZeppelinOS's user base to reach a wider audience. In exchange, developers commit to the challenge mechanism, by which their vouched tokens can be reduced if the package is deficient (see Figure 1).

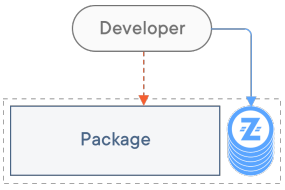

Figure 1: Publishing New Packages

1.2 Challenging Packages

Challenges to packages involve bug reports as well as reporting dishonest behavior like cloning or integrating rejected contributions, and require vouching an amount of ZEP in correspondence with the severity of the challenge. Package developers can willingly accept a legitimate challenge, giving a fraction of the package vouched tokens to the challenger, or reject it. In the latter case, a vote among ZEP holders is called: if accepted, the package's vouch is doubly reduced: part goes to the challenger and part to the voters that supported this outcome. If rejected, the challenger loses the vouched tokens, which go to the reviewers that voted against the challenge. The challenging mechanism ensures only high quality packages are available through ZeppelinOS (see Figure 2).

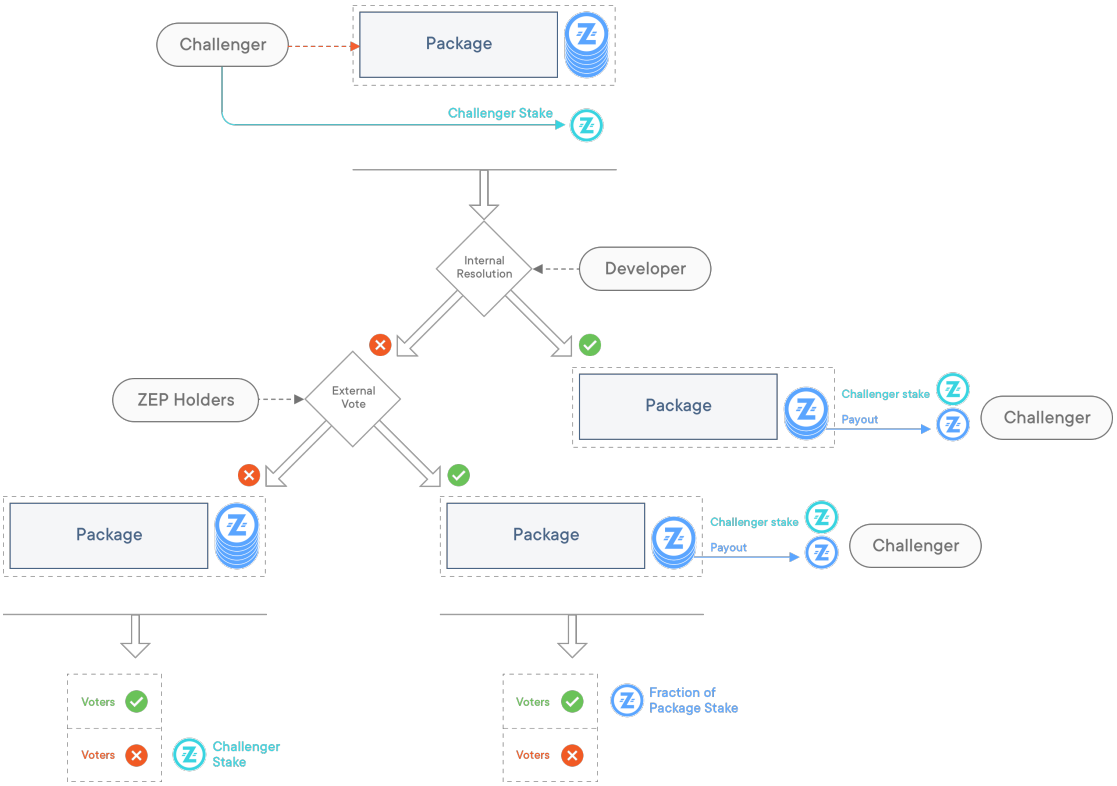

Figure 2: Challenging Packages

1.3 Contributing Code

Code contributions to packages do not require vouching, but simply a claimed reward. Developers can again choose to either accept the contributions and reward the authors using the package vouched tokens, or reject them. In this last case, the contributor is free to publish a new package that incorporates the changes (see Figure 3).

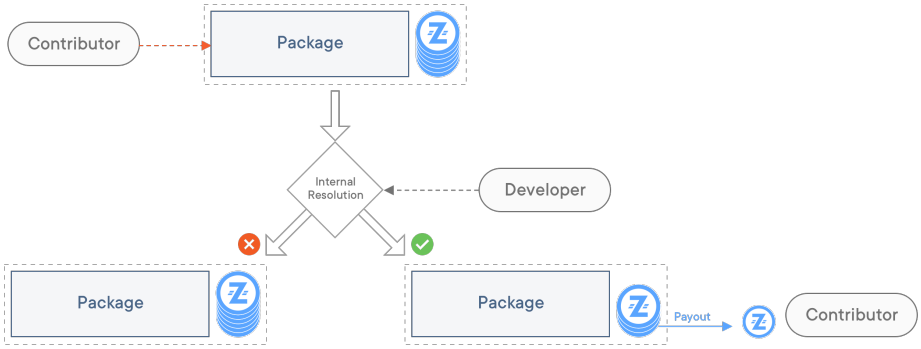

Figure 3: Contributing Code

1.4 Supporting Packages

Any actor in the network can vouch towards a package to show support. This vouch is equivalent to the original developer's vouch, and will be used to pay out challengers and contributors. In retribution, supporters get a fraction of the linking fees, as determined by the package developer in their monetization model. This mechanism provides a way for ZeppelinOS token holders to actively participate in the system in a way that is aligned with their typical expertise (see Figure 4).

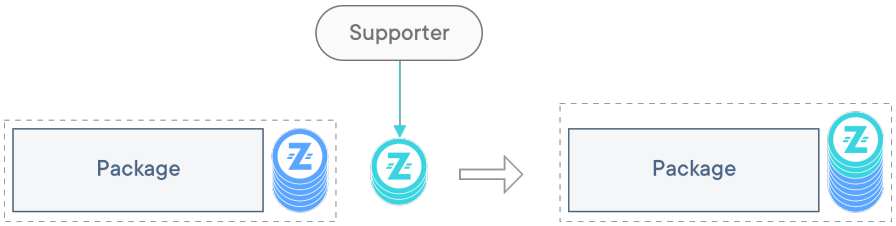

Figure 4: Supporting Packages

1.5 Linking Packages

Developers using ZeppelinOS can use the available packages in their projects. If the packages have a linking fee set up, users will need to pay this fee in order to link the packages through the system (see Figure 5).

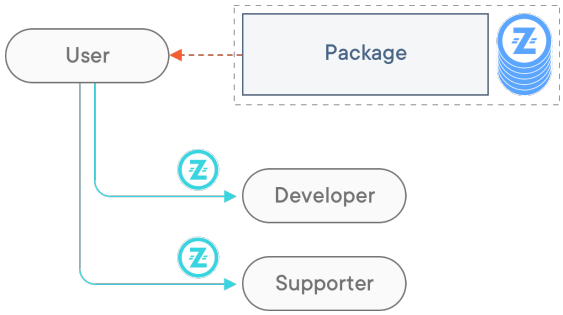

Figure 5: Linking Packages

2 Other ZeppelinOS Components

Apart from being at the core of the EVM package system mechanics, ZEP has other, simpler uses in the system.

2.1 Marketplace

The marketplace will be a place where users can acquire services for their projects such as file storage, distributed computing, or oracle provisions. While service providers normally have their own tokens, ZeppelinOS will provide a way to purchase these services using ZEP in a way that is transparent to the users, so that they don't need to handle more than one currency.

2.2 Scheduler

The scheduler will enable scheduling transactions for cases in which they need to be triggered manually, or when the gas needs to be paid for by someone other than the soliciting user. Its cost will be paid in ZEP.

2.3 System Governance

Finally, the entire ZeppelinOS architecture will be subject to upgrades when needed. These upgrades will be controlled by the ZEP stakeholders.

For a more in-depth explanation and to understand the process that led to this design, please refer to Part I, II and III of the ZEP token dynamics design series.