aelf Economic and Governance Whitepaper

V1.2.0

October 25th 2022

CONTENT

1. Overview

1.1 Economic Model Overview

This document describes the various roles within the aelf ecosystem, outlining the incentives, rights, and responsibilities for production nodes, candidate nodes, developers, and token holders.

The aelf economic model plays an important role in strengthening the underlying ecosystem, ensuring the overall stability and functioning of the network, and encouraging community collaboration to promote sustainable and healthy development.

1.2 Definition of Roles in the System

In aelf's ecosystem, there are the following roles: production nodes, candidate nodes, developers, and token holders (users).

- **Production node**: Any person/organisation can apply to become a production node. If successful, they will participate in block production according to the consensus rules and receive block rewards.
- **Candidate node**: Any person/organisation who is not a production node according to the vote ranking.
- **Developer**: An individual or organisation that develops applications on the aelf network.
- **Token holder (user)**: An individual or organisation who owns any aelf platform valuable assets and plans to use the services offered by developers on the aelf network.

2. Token Model

2.1 Types of Tokens

There are 4 types of token: ELF token, resource tokens (CPU resources, RAM resources, DISK resources, NET resources, READ resources, WRITE resources, STORAGE resources, TRAFFIC resources), tokens created by developers, VOTE Token and SHARE Token.

- **ELF**: The main token on the aelf platform, used for transaction fees, SideChain index fees, production nodes deposits, voting, and block rewards.
- **Resource Token**: This token is used by developers to pay for the consumption of resources when chains or DApps are running. Developers need to ensure that there are sufficient resource tokens for their application's normal requirements. Resource tokens can be bought and sold for ELF tokens.
- **Tokens created by developers**: On the aelf platform, developers can create tokens and build their own token models and incentive mechanisms.
- **VOTE Token**: Users obtain voting rewards by voting for production nodes. After the vote, the VOTE token will be used to redeem the vote. SHARE token is used to obtain a SideChain rewards.

2.2 Issuance and Release Plan

- Total ELF issuance: 1 billion
- Mining ratio: 120 million tokens (12%)
- Block Reward Release plan: halved every four years
- Total supply of ELF and resource tokens is fixed
- Total resource tokens issuance: 500 million each CPU resources tokens, RAM resources tokens, DISK resources tokens, NET resources tokens, READ resources tokens, WRITE resources tokens, STORAGE resources tokens, and TRAFFIC resources tokens.

2.3 Burning Mechanism

The following two scenarios will result in the burning of a certain proportion of ELF:

- **Transaction fee**: 10% of the transaction fee is burned, and the remaining 90% of the MainChain goes into the rewards pool. If the fee receiver is set on the SideChain, the remaining 90% will be sent to the address, otherwise, all will be burned.
- **Resource purchasing**: Purchasing resource tokens incurs a 0.5% fee, 50% of which are destroyed and 50% enters the rewards pool

2.4 Resource Token Trading Model

Resource token transactions are using the Bancor model. There is no inquiry process as buying and selling prices are determined by the Bancor model. It can guarantee users an efficient and fast token trade, even with a small number of users and a small transaction volume, the instant exchange of resource tokens and ELF can be guaranteed.

When issuing resource tokens, the total price is anchored to a fixed ELF price:

$$cw(\textit{connector weight}) = \frac{\textit{connector balance}}{\textit{resource token supply} \times \textit{price}}$$

cw is a fixed value, connector balance is the total amount of anchored ELF
When buying and selling an amount of $\mathrm{d}S$ resource tokens, assume:
cw is F; connector balance is B; resource token supply is S;
the price of the resource token is P, this will give the following equation:

$$B = FSP \qquad (1)$$

$$P\mathrm{d}(S) = \mathrm{d}(B) = \mathrm{d}(FSP) = F(P\mathrm{d}(S) + S\mathrm{d}(P)) \qquad (2)$$

$$\frac{\mathrm{d}P}{\mathrm{d}S} = \frac{1-F}{F} \times \frac{P}{S} \qquad (3)$$

$$\ln(P) = \frac{1-F}{F} \times \ln(S) + C(constant) \qquad (4)$$

The relationship between P and S at each time can be determined. Therefore, at any given time, buying/selling $\mathrm{d}S$ resource tokens by spending/receiving ELF can be calculated by $\int P\mathrm{d}S$. For example, when purchasing ΔS resource tokens, you need the following amount of ELF:

$$= \int_{S}^{S+\Delta S} P\,\mathrm{d}S = FP_0S_0\left(\left(1+\frac{\Delta S}{S_0}\right)^{\frac{1}{F}} - 1\right) = B_0\left(\left(1+\frac{\Delta S}{S_0}\right)^{\frac{1}{F}} - 1\right) \qquad (5)$$

At the same time, the inverse operation of formula (5) can be used to obtain how many resource tokens can be exchanged for a specified amount of ELF:

$$\Delta S = S_0\left(\left(1+\frac{\Delta B}{B_0}\right)^{F} - 1\right) \qquad (6)$$

If there is an intermediate token, such as the Relay token, it will anchor both ELF tokens and resource tokens. If a user wants to use a number of ΔB ELF tokens to buy a certain number of ΔS Relay token, then, use those ΔS Relay tokens to buy some resource tokens. The relationship between resource token and ELF token is:

The ELF spent (obtained) is calculated by the following formula (*a* is the variable for users to purchase/sell tokens):

Purchasing:

$$\text{cost} = bf \times \left(\left(\frac{bt}{bt - a} \right)^{\frac{wt}{wf}} - 1 \right) \tag{7}$$

Selling:

$$gain = bt \times \left(1 - \left(\frac{bf}{bf + a} \right)^{\frac{wf}{wt}} \right) \tag{8}$$

The following constants are determined by a parliamentary vote:

- bt= balance of to connector (the real-time position of the to the connector is initially determined by the production nodes)
- bf= balance of from connector (the real-time position of the from the connector is initially determined by the production nodes)
- wt= weight of to connector (weight constant)
- wf= weight of from connector (weight constant)

a is passed in when a user purchases/sells a Token, which indicates the number of Tokens to be purchased/sold. When purchasing, the larger the *a* value is, the more ELF is spent per token. When selling, the larger the *a* value is, the lower the unit token price is.

Using this transaction model, from the perspective of the economic model, the value of both connectors is an equivalent amount of Relay tokens at any given time.

Assume that the resource Token connector balance is initially set to A and the ELF token connector balance is initially set to M. When the number of N ELF tokens spent on purchasing a set number of *a* resource token is put into a mortgage account:

- Resource Token connector balance is (A - a)
- ELF token connector balance is (M + N)

According to the model, the value of two tokens is the same at any given moment, $V_{A-a} = V_{M+N}$, It can be seen that when *a* is infinitely close to *A*, the ELF price corresponding to the remaining resource tokens approaches infinity. Since this equation holds consistently, the more resource tokens being purchased, the more ELF tokens will be needed. On the contrary, when users sell more resource tokens, the more ELF tokens they will receive. This feature will also help to avoid speculation.

2.5 Pay-per-time Model

In the process of Internet application development, developers usually rent cloud service as their servers to deploy their applications, and then provide services to users. The application can utilize the server's computing resources exclusively. In aelf, SideChain developers can benefit by using a single SideChain by paying CPU Token, RAM Token, DISK Token, and NET Token. This SideChain has independent computing and storage resources. The exclusive SideChain is priced according to the time used. The unit price of the fee is determined through negotiation between the production node and the developer.

2.6 Cross-chain Index Fee Model

SideChain developers who want to implement cross-chain transfers and cross-chain verification need to have the MainChain index the SideChain's block information. In order to use it, they need to pay the block index fee. The unit used for the block index fee is ELF. The amount charged is determined conjointly by the organisation and the developer. The initial index fee is passed in as a parameter when applying to create a SideChain. The index fee amount can be adjusted through a proposal. It will take effect when both the organisation and the developer agree to the adjusted plan.

2.7 Revenue Sharing Model

In order to make SideChains easy to use, aelf enables developers to use SideChain resources through revenue sharing. This way, developers must promise to share a certain percentage of their revenue (the specific number is co-determined by the developer and parliament members), and deploy a Token Holder contract on the SideChain.
Users who vote on the MainChain will receive SHARE Token (1:1). They can transfer these tokens to the SideChain for staking in the Token Holder contract and receive SideChain rewards.

2.8 Transaction Fee Model

There are two types of charging models in the aelf system: Developer Charging Model and User Charging Model. When developing contracts, developers can choose the Developer Charging Model or User Charging Model.

2.8.1 Developer Charging Model

The Developer Charging Model assigns the developer to cover the fees for the execution of transactions. Unlike the developer Pay-by-time model, the Developer Charging Model is a pay-on-demand method. This model collects developer transaction fees by analyzing the use of resources during transaction execution. Resource tokens are used to measure the on-demand use of resources by DApps. These resource tokens include: READ token, WRITE token, STORAGE token, and TRAFFIC token. Among them, the READ/WRITE tokens are used to pay for the read and write of the state database when the transaction is executed (the CPU and RAM are used on demand). The transaction size will affect the network transmission and final storage, the STORAGE token is used to pay for storage resources used on-demand, and the TRAFFIC token is used to pay for network bandwidth resources, also used on demand.

Each resource is given a threshold value according to its use in transactions. This threshold is used to determine the normal charging interval, as well as the adjusted charging interval when the threshold is more or less than the normal threshold. In the normal charging interval, charges are based on the unit price of the resource used. In addition to extra charging intervals, the additional power consumption is calculated based on the squared level of the excess. The establishment of specific thresholds is calculated based on the use of various resources in actual transactions.

- READ Token:

$$token \text{ cost} = \frac{count}{8} + 100000 \qquad count \le 10$$

$$token \text{ cost} = \frac{count}{4} \qquad 10 < count \le 100$$

$$token \text{ cost} = \frac{25}{16} count^2 + \frac{1}{4} count \qquad count > 100$$

Count: the basic unit for measuring the consumed READ resources. Variables are calculated by the program in real-time based on actual transactions. Among them, 100,000 is because the calculation accuracy is 8 bits, which actually represents a constant of 0.001. The coefficient of the formula can be updated by upgrading the contract.

- WRITE Token:

$$token\ \text{cost} = \frac{count}{8} + 100000 \qquad count \le 10$$

$$token\ \text{cost} = \frac{count}{4} \qquad 10 < count \le 100$$

$$token\ \text{cost} = \frac{25}{16} count^2 + \frac{1}{4} count \qquad count > 100$$

Count: the basic unit for measuring the consumed WRITE resources. Variables are calculated by the program in real-time based on actual transactions. Among them, 100,000 is because the calculation accuracy is 8 bits, which actually represents a constant of 0.001. The coefficient of the formula can be updated by upgrading the contract.

- STORAGE Token:

$$token\ \text{cost} = \frac{count}{4} + \frac{1}{100000} \qquad count \le 1MB$$

$$token\ \text{cost} = \frac{count^2}{2000} + \frac{count}{64} \qquad count > 1MB$$

Count: the basic unit for measuring the consumption of STORAGE resources. Variables calculated by the program in real-time based on actual transactions. Among them, 100,000 is because the calculation accuracy is 8 bits, which actually represents a constant of 0.001. The coefficient of the formula can be updated by upgrading the contract.

- TRAFFIC Token:

$$token\ \text{cost} = \frac{count}{64} + \frac{1}{100000} \qquad count \le 1MB$$

$$token\ \text{cost} = \frac{count^2}{2000} + \frac{count}{64} \qquad count > 1MB$$

Count: the basic unit for measuring consumed TRAFFIC resources. Variables calculated by the program in real-time based on actual transactions. Among them, 100,000 is because the calculation accuracy is 8 bits, which actually represents a constant of 0.001. The coefficient of the formula can be updated by upgrading the contract.

2.8.2 User Charging Model

The User Charging Model refers to the costs incurred by the user during the execution of the transaction, which consists of two parts. One part is charged according to the size of the transaction and the other is set by the developer in the contract. The formulas are as followings:

$$token\ \mathrm{cost} = \frac{x}{800} + \frac{1}{10000} \qquad count \leq 1MB$$

$$token\ \mathrm{cost} = \frac{x}{80} \qquad 1\ MB \leq count \leq 5MB$$

$$token\ \mathrm{cost} = \frac{count^2}{100000} + \frac{count}{80} \qquad count > 5\ MB$$

Count: the basic unit for measuring the size of transactions executed. Variables are calculated by the program in real-time. Among them, 100,000 is because the calculation accuracy is 8 bits, which actually represents a constant of 0.001. The coefficient of the formula can be updated by upgrading the contract.

3. Incentive Model - MainChain

3.1 Rewards Pool

aelf distributes rewards through rewards pool, and the bonus of the MainChain rewards pool is composed of the following parts:

- Block Reward:
  The aelf foundation has issued 12% of the total ELF tokens in accordance with the white paper for community governance rewards, with a portion given to Production Nodes based on the number of blocks produced. The reward amount is reduced by 50% every 4 years. The aelf blockchain produces 8 blocks every 4 seconds, with each block initially generating 0.125 ELF;

Time	**Reward**
1st to 4th year	0.125 ELF for each block produced
5th to 8th year	0.0625ELF for each block produced
9th to 12th year	0.03125 ELF for each block produced

- Transaction Fee:
  Transaction fee should be paid when sending transactions on the MainChain. Meanwhile, the remaining 90% of the transaction fee goes into the rewards pool. On the SideChain, if the fee receiver is set, 90% of the transaction fee will be sent to the address. Otherwise, all will be burned.

- Resource Tokens Transaction Fee:
  Trading resource tokens will generate transaction fees, and all the transaction fees generated will enter the rewards pool.

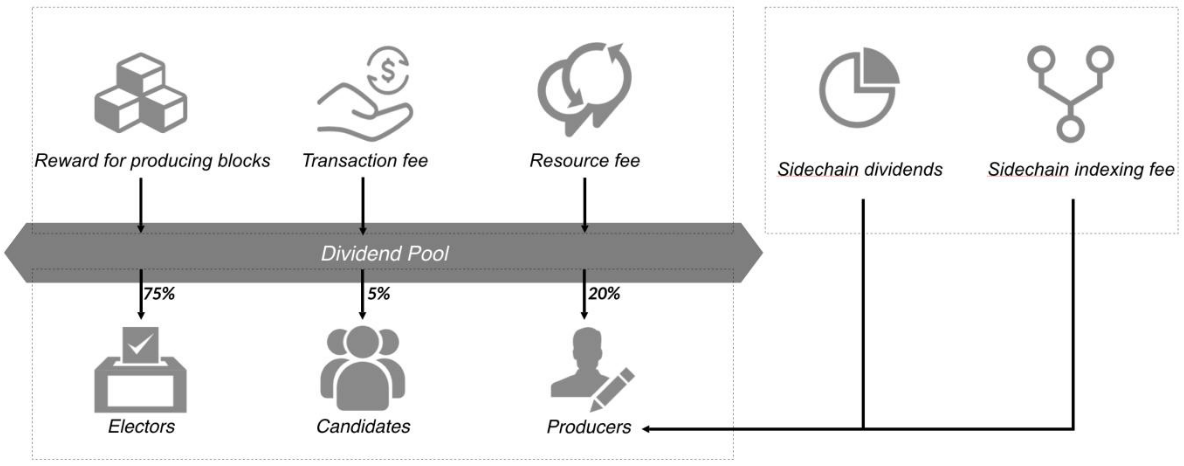

Figure 1.3 Rewards Pool Structure

Distribution method and proportion of rewards pool:

Types	**Percentage of the Rewards Pool**
Production Node Basic Rewards	10%
New Production Node Rewards	5%
User Rewards from New Node	5%
Candidate Node Rewards	5%
Voter Rewards	75%

3.2 Production Nodes Incentives

Production Nodes Responsibilities:

1. Collect, verify, and package transactions into blocks, broadcasting the blocks to other nodes;
2. After verification, the block will be added to the local blockchain to maintain the stability of the entire aelf ecosystem.

Production Nodes Incentives:

1. The number of blocks produced in the previous session is used as the distribution profit weight for the production nodes, the "basic rewards of production nodes" part of the rewards pool (accounting for 10% of the rewards pool);
   Note: The weight calculation method is as follows: if the number of blocks produced by a certain production node is greater than or equal to 80% of the average, the actual number of blocks produced during the trial period will be used as the weight; if the number of blocks produced by a certain production node is smaller than 80%, the rewards weight will be reduced proportionally; if the number of blocks produced by a production node is less than 50% of the average number of blocks produced, its weight is 0, that is, the production node cannot claim rewards.
2. If there is a new elected node in this session, the new node will equally share the "new node rewards" (worth 5% of the rewards pool; otherwise, this part of the rewards will be merged into the "production node basic rewards".
3. If there is a new elected node in this term, the "flexible rewards" (worth 5% of the rewards pool) will merge into the "voter rewards"; otherwise, this part of the rewards is merged into the "production node basic rewards".
4. Benefits from building a SideChain network for developers:
   a. Index fee rewards for indexing blocks for SideChain developers;
   b. Collect resource fees from SideChain developers;
   c. The rewards brought by the SideChain support plan.

   Note: Item 4 is described in more detail in the "Incentive Model-SideChain Developers" section.

3.3 Candidate Node Incentives

Responsibilities of candidate nodes: Verify block, alternate production nodes, and maintain the stability of the entire aelf ecosystem.

Incentives: According to the number of votes obtained (the number of production nodes * 5), the "candidate node revenue" portion is evenly distributed (5% of the rewards pool).

3.4 Voters Incentives

Voters responsibilities: vote for production nodes to represent the will of the community

Incentives: The incentive proportion is allocated according to the weight of the number of users voting and the lock period specified. To vote for a node, users need to choose the period of time for which they will lock their votes. Users can choose 3, 6, 12, 24, or the maximum lock period of 36 months. The distribution will be based on the principle of longer lock periods for a bigger reward.

The ratio weight of the lock period to the number of votes is 2:1

The weight of each vote is determined by both the number of tokens and the duration of the lock-up. The growth rate of daily weight differs for different time spans:

- The growth rate of daily weight within 1 year: 0.1%
- The growth rate of daily weight within 2 years: 0.15%
- The growth rate of daily weight for more than 2 years: 0.2%

The weight is calculated according to the following formula:

$$total\ weight\ = amount\ \times ((1\ +\ growth\ rate)^{t}\ +\ k)$$

amount is a variable number of user votes, the variable *t* is the vote corresponding locking time (in days), the two variables determined by users.

All users who vote can calculate their rewards according to the above formula. They will become assigned users who received their portion of the rewards pool. (75% of the rewards pool)

Parliament organisations can change the following constants in the form of the proposal:

Growth rate of daily weight: the corresponding time period and growth rate can be customized. For example: (30, 0.01), (60, 0.02) means that 0 to 30 days is calculated as 0.01, and 0 to 60 days is calculated as 0.02.

k: Indicates the weight ratio between the number of votes and the voting times. The default value is 1/2.

3.5 Number of Nodes

According to the aelf Whitepaper, during the initial phase, 17 production nodes will be elected from candidate nodes. After the chain operates robustly, 2 new production nodes will be added each year. The maximum number of production nodes will be determined by the community after voting on the Mainnet to meet the needs of ecosystem development.

3.6 Production Node Elections

3.6.1 Election Requirements

- The number of ELF tokens needed to be deposited by a node candidate: ≥100,000 ELF.
- Configuration recommendations: 8-core processor, 16GB memory, 500GB hard drive, sufficient bandwidth to ensure block and transaction synchronization.
- Node geographical location: worldwide
  Note: Deposited tokens will be locked after successful election to a production node. The locked tokens will be returned to the node owner upon them stepping down from running the production node, assuming there is no violation of operation.

3.6.2 Election Process

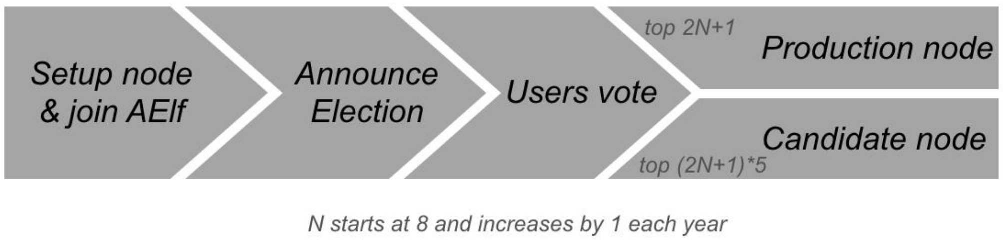

Figure 3.7 Node Election Procedure

1. Teams or individuals who are interested in becoming a production node can join the election after successfully building a node and connecting it to the aelf Block Explorer
2. A list of candidate nodes will be announced to the community and the top 17 nodes by votes will be elected to production nodes.
3. The production nodes exist within a rotation system, and a new round of production node elections will be held every week. Nodes will be listed from highest to lowest based on the number of votes received.

3.7 Voting Rules

Voters can participate in node elections using ELF with the ratio of one ELF token equal to one vote. There are no restrictions on the number of votes an individual may cast or the number of nodes they may vote for.

1. The default locked period for voters is 3 months and users cannot withdraw tokens during the locked period without losing their rewards.
2. Users can freely choose the voting locking period. The locking period is set at 3 months, 6 months, 12 months and 24 months by default, with a maximum of 36 months (each month is calculated as 30 days);
3. During the locking period, users can freely change which node receives their votes.
4. Voting rewards will vary based on the number of votes cast and the locked time selected. The longer the locked period, the higher the rewards will be.
5. After the locking period ends, locked tokens can be redeemed (deemed as abandoning votes), and will no longer earn rewards.

3.8 Production Node Elimination Mechanism

3.8.1 Election Period

A new round of production node elections will be held every week and the current production node automatically enters the next round of elections as long as deposits are still locked.

3.8.2 Elimination Rule

If a production node does not produce any block in 72 hours, the qualification of the production node will be automatically cancelled and the deposited ELF will be destroyed.

If a production node packages a malicious transaction, other nodes will fail to verify the block they receive and will disconnect the network connection with the production node that packaged the malicious transaction. This production node will also be added to the blacklist.

3.9 Rewards on Token Holder Contract

On aelf, developers can issue tokens for their contracts, and users who hold these tokens can enjoy the rewards of the corresponding contract. The specific steps are as follows:

1. Use the Token Holder contract in the developer contract to create a token holder reward plan;

2. In the developer contract, the promised reward is entered into the reward scheme through the method provided by the Token Holder contract;
3. The user holding the corresponding token obtains the rewards weight from the reward plan through locking;
4. When developers collect rewards, they will automatically release a portion of the rewards, and users who participate in the lockup can receive rewards.

4. Incentive Model - SideChain

aelf's 'MainChain + multi-SideChains' model is core to the overall ecosystem with each SideChain adopting a "one chain, one scenario" design. The purpose of this design is to ensure effective segregation of resources and to provide the platform with scaling capabilities. This section describes the incentive model for SideChain developers.

4.1 Rewards Pool Mode

If the developer chooses the revenue sharing payment model and promises to allocate a certain proportion of income to the SideChain ecosystem, then this part of the cost will enter the SideChain rewards pool. Users holding SHARE Token can share the SideChain rewards by staking the SHARE token to the SideChain rewards pool. The rewards pool is allocated according to the user's staking ratio.

4.2 SideChain Creation

When developers create an application that requires independent computing resources to ensure operating efficiency, it is necessary to consider creating a separate SideChain for it.

When users need to increase/decrease the performance of the SideChain, they can apply to the system. This process is as convenient as a cloud service provider's elevator server, avoiding unnecessary waste of resources.

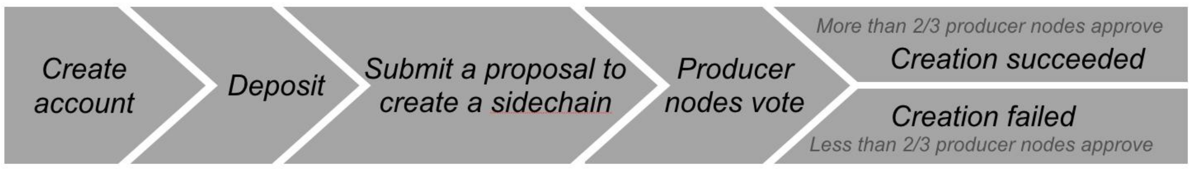
Create account
Deposit
Submit a proposal to create a sidechain
Producer nodes vote
More than 2/3 producer nodes approve
Creation succeeded
Creation failed
Less than 2/3 producer nodes approve

Figure 4.1 SideChain Creation Procedure

1. Create an account
2. Deposit a certain amount of ELF in the account; (the specific amount is agreed on by the production nodes)
3. Call the contract to initiate the creation of a SideChain application
4. Production node voting (need to receive more than 2/3 of the production node votes)
5. If the production nodes agree to the request to create a SideChain, the creation of the SideChain node will be completed within 3 days.

After the SideChain is successfully created, the deposited ELF will be used to deduct the index fee (the index fee amount is jointly determined by the production nodes). Developers need to ensure that the account balance is sufficient so that the SideChain can be indexed normally.

4.3 SideChain Closing Mechanism

When an independent SideChain developer needs to close a SideChain for any reason, they must apply for review by the production nodes. Upon review, the production nodes have the ability to complete the SideChain closure.

When the balance of the index fee account on the SideChain is insufficient, the MainChain will automatically stop indexing on the SideChain. If this occurs, the production nodes have the right to stop the services of the particular SideChain.

Risk warning:

- Stopping the index function means that cross-chain transfers and cross-chain verification cannot be performed, and users' tokens on this SideChain cannot be transferred.

4.4 SideChain Developer Charging Model

4.4.1 Exclusive SideChain Charging Model

The charging model of an exclusive SideChain is as follows:

1. The pay-per-time model pays the exclusive resource usage fee (CPU resource / RAM resource / DISK resource / NET resource) according to the actual block production time. This is like a developer who needs to deploy an application to a

cloud server. Pay according to the configuration of the cloud server (CPU / RAM / DISK and fixed bandwidth);

2. The cross-chain index fee model, in order to ensure cross-chain transfer/verification capabilities, a certain amount of ELF needs to be mortgaged to pay the index fee when creating a SideChain. The index fee is charged according to the number of blocks, and the unit price is determined through consultation between the parliamentary organisation and the developer. The MainChain production node performs a SideChain node indexing, allowing the SideChain to have cross-chain transfer and verification capabilities, extending the functions of the SideChain and maintaining the security of the SideChain network. The MainChain production node obtains the index fee for this.

An exclusive SideChain allows developers to choose the transaction fee model and set the transaction fee price and the transaction fee receiving address.

4.4.2 SideChain Support Plan Charging Model

Proposal for SideChain developer support plan to waive initial pay-per-time model

1. Set up 4 SideChains to support the innovation of exclusive SideChain developers, using the following fee model:
    a. Developers Sharing Model, developers promise to share a certain percentage of income to production nodes;
    b. Cross-chain index fee model (same as the exclusive SideChain fee model)
2. Developers applying for the "Support Plan" need to submit a project plan and sharing plan, which will be reviewed by the production nodes. After passing the review, they can utilize the "SideChain Support Plan" for three months;
3. After three months, the production node will vote to continue or stop support;
4. After support is stopped, the production node has the right to stop the services of this SideChain.

4.4.3 Sharing SideChain Charging Model

SideChain Sharing Proposal: Set up a shared SideChain which any developer can deploy contracts on. The shared SideChain can choose to use the developer paying transaction fee model or developer revenue sharing model. The contract can be deployed after the contract is approved by the parliamentary.

Developer-pay-transaction-fee model: Developers have to pay for the resources they consume. This will enable developers to optimize the program as much as possible to reduce the consumption of resources. Transaction execution for this contract fails when the resource Token is exhausted. The developer needs to hold sufficient resource tokens to keep the contract running. Developer transaction costs will go into the shared SideChain consensus contract to be evenly distributed to the production nodes.

Developer revenue sharing model: Developers distribute part of the revenue to the SideChain rewards pool under the contract implementation. Users in the rewards pool can benefit from this.

The creator of the shared SideChain has to pay a cross-chain index fee to ensure cross-chain transfer and verification functions.

4.5 Production Nodes Incentives

4.5.1 Responsibilities in SideChains

- Construct and maintain SideChain nodes allowing developers' applications to run stably
- Maintain the security of the SideChain

4.5.2 Rewards in SideChain Ecosystems

- The block index fees are paid by developers in units of blocks. The unit price of an index fee varies according to the type of resource selected by the SideChain and is agreed by the production nodes.
- Resource fees paid by developers to ensure the normal operation of the application. (regardless of whether they participate in the SideChain support plan, there will be transaction fees).
- If developers use the SideChain support plan, the developer should promise to contribute part of the profit to the SideChain ecosystem. The production node can stake SHARE Token to share the profits.

- Developer payment transaction fee model: transaction fee will directly send to the transaction fee receiving address, which will be allocated to the production node when the SideChain indexes the MainChain data.

5 Governance

Governance is very important for the entire community and is the foundation for ensuring the sustainable and healthy operation of the community. aelf provides three governance models. Through these governance tools, users, candidate nodes, and production nodes can fully participate in the governance of various affairs and actively play their respective roles to make the whole ecosystem more just.

These three governance models have played an important role in the governance of aelf's MainChain and SideChains, and are also powerful tools for SideChain developers to build their own governance systems by combining their own scenarios.

Governance is achieved through the form of organisation, which is the carrier of governance authority. Governance rights can be transferred to other organisations with the consent of the organisation, and the new organisation will take over the old organisation to exercise governance rights.

5.1 Governance Model

5.1.1 Parliament Governance Model

Users vote for production nodes that exercise transactional governance rights on behalf of most users. In the aelf network, the production nodes use the Parliament model to govern important matters. In the initial state, there is only one genesis parliamentary organisation generated by the system.

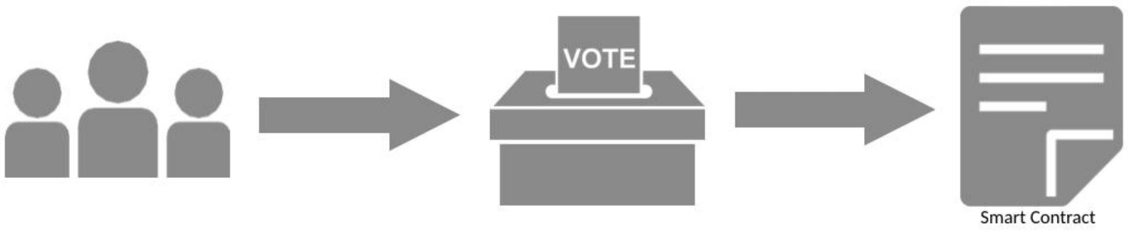

Figure: 5.1.1 Parliament Governance Model

Members of Parliament can follow these steps to create and use a new Parliamentary organisation:

1. Create a parliamentary organisation and set the conditions that trigger decision execution (members are production nodes);

    Examples of trigger conditions: more than 2/3 of the production nodes voted yes, less than 1/5 nodes voted against, and less than 1/5 nodes voted absent, and the number of participants was not less than 3/4.
2. Members of parliament organisations expressed their opinions: in favour, against or abstaining;
3. When the set conditions are reached, the execution of the decision is triggered.

5.1.2 Association Governance Model

In addition to the whole community, which is a large organisation, there will be many small organisations on the aelf platform. These organisations were established in order to achieve certain goals, and they also need effective governance tools to cooperatively handle transactions within the organisation, such as: DAO (distributed Autonomous organisations) and so on. aelf provides an association governance model that specifically addresses governance within the association.

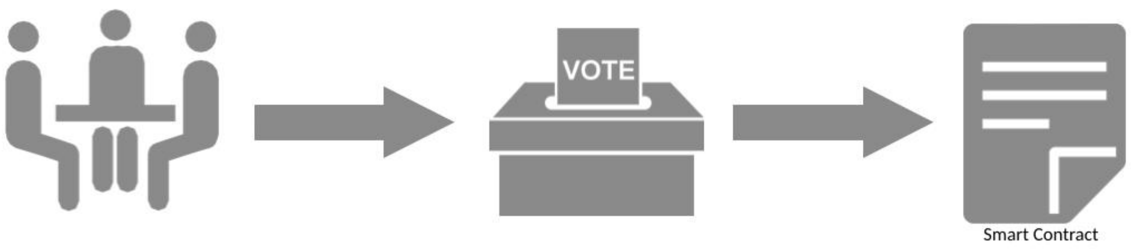

Figure: 5.1.2 Association Governance Model

Use Association Governance as follows:

1. Create association organisations, add members of association organisations and set conditions that trigger transaction execution.

    Examples of trigger conditions: There are 5 members in the association, only more than 3 members vote in favour, less than 2 members vote against, less than 2 members abstain from voting, and at least 4 members participate to trigger transaction execution.
2. The members of the association expressed their opinions: yes, no or abstention;
3. When the set conditions are reached, the execution of the transaction is triggered.

5.1.3 Referendum Governance Model

Production nodes or associations cannot determine all decisions. Some extremely important decisions, especially those involving user rights and interests, should involve all users and give full control to the user's voting for governance. The Referendum governance model is built for this.

There are two referendum governance models in aelf. The first is to execute the decision after meeting the set conditions (vote / no / abstain); the second is to set options, and users can choose options according to their wishes.

The first referendum model is shown below:

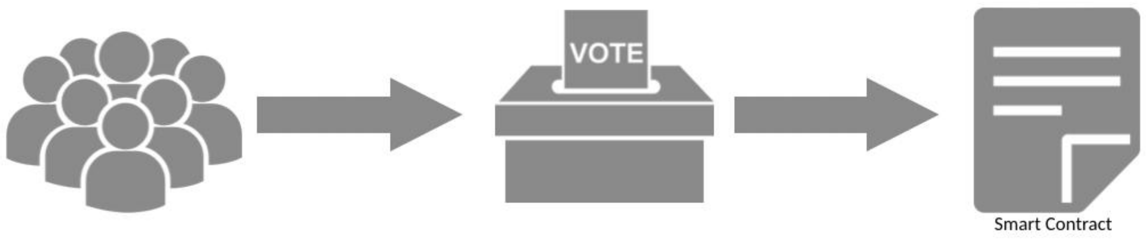

Figure: 5.1.3-1 Referendum Governance Model One

Follow the steps below to use the Referendum governance model:

1. Create a referendum organisation, set the referendum token type, and trigger conditions (the number of mortgage tokens);

    Examples of trigger conditions: The total number of votes is not less than 1,000,000, the votes in favour are not less than 500,000, the votes against are less than 100,000, and the number of abstentions is less than 100,000.
2. Call on the public to use the set Token to participate and express their opinions: yes, no, or abstain;

3. When the set conditions are met, the execution of a specific decision is triggered.

The second referendum model is shown below:

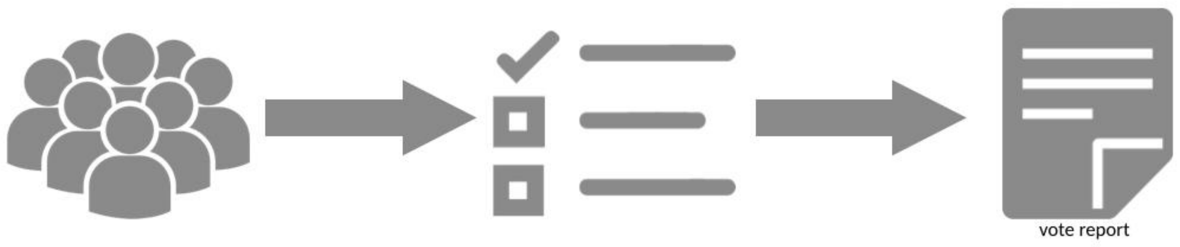

Figure: 5.1.3-2 Referendum Governance Model Two

Follow the steps below to use the Referendum governance model:

1. Create voting items and set the referendum options: the type of Token used for voting, whether it is locked, the start and end time of voting, etc.;
   Voting example: Set the token type to VOTE, lock to true, the voting start time is now, and the end time is one week later. When the time is up, voting will end automatically;
2. Call on the people to vote on the referendum options;
3. After the vote expires, the vote is completed.

5.1.4 Customized Governance Model

In addition to the three governance models mentioned above, in order to meet more complex governance scenarios, aelf provides the ability to cascade governance, connecting multiple of the same or different types of governance models in series or in parallel to form complex customized governance models. Decisions can only be performed if every organisation agrees.

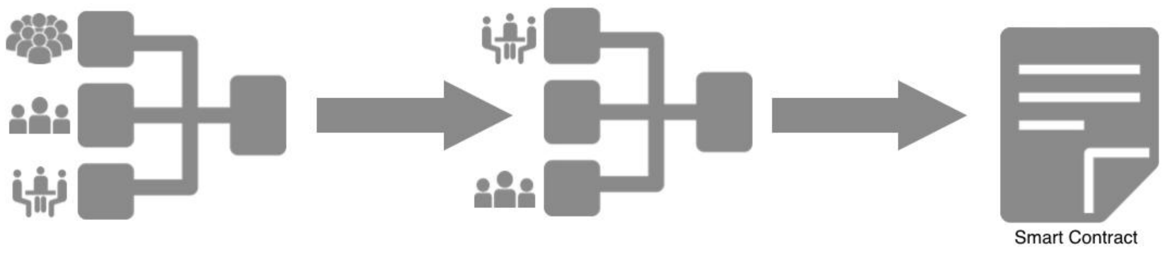

Figure: 5.1.4 Custom governance model

Follow these steps to use a complex governance model:

1. Create their own organisations (parliament organisations, association organisations, referendum organisations) based on actual governance;
2. According to the actual situation, various organisations are connected in parallel or in series;
3. Call on members of relevant organisations to participate in governance to express their opinions;
4. When the set threshold is reached, the final decision execution is triggered.

5.2 Current MainChain Governance

Three basic governance models make the governance of the MainChain easy. This section describes how to use these three basic governance models to govern MainChain transactions.

5.2.1 Using the Parliament Governance Model

Examples of triggering conditions for the parliament governance model: in parliament organisations, only when the votes in favour exceed 2/3, the votes against it are less than 1/5, the abstentions are less than 1/5, and the number of votes is not less than 3/4 will the execution of a decision be triggered.

The affairs governed by the parliament model include but are not limited:

1. MainChain contract deployment / upgrade
   Although there are automatic contract inspections, for security, the MainChain deployment and upgrade contracts need to be deployed/upgraded after approval by the parliament during the initial stage of the launch. (This is also the governance process of the MainChain DApp)
2. Create a connector between the third-party token and ELF
   In aelf, by staking ELF, you can create connectors with ELF, and complete exchange with ELF. This rule is based on the Bancor model. Tokens that have established and enabled connections will follow the Bancor model for transactions. Operations that request the creation and activation of connectors use parliamentary organisation governance.
3. Adjust the reward rate for user votes
   Users who vote will receive rewards from the network, subject to the vote lock-in time. Longer periods result in higher rewards. The estimated daily rewards for one year, two years and three years can be adjusted by the parliament. For specific adjustable parameter range, please refer to 3.4.

4. Adjust system contract method fee
   The system contract transaction fee consists of two parts: one is charged according to the size of the transaction, and the other part as a contract method fee is set by the parliament.
5. Configure the token list that can be used for transaction fee
   Transaction fees can be paid using not only ELF but also other selected tokens (such as stablecoins). A list of tokens that can be used can be configured and is co-managed by parliament.

6. Weight Adjustment in rewards pool and modification for supported token list
   The bonus ratio of the rewards pool to each item can be adjusted, and the token list supported by the rewards pool can be modified.

5.2.2 Using Association Governance Model

The association is an autonomous organisation, and the affairs within the association can be determined in collaboration through association governance, such as:

1. Association members jointly manage account assets
   Example of association governance model trigger conditions: There are 5 members in the association, only more than 3 members vote in favor, less than 2 members vote against, less than 2 members abstain, and at least 4 members can move the organisation The balance of the account.

5.2.3 Using the Referendum Governance Model

The affairs governed by the referendum governance model include but are not limited:

1. Election of production nodes and candidate nodes
   Users vote for the production nodes and candidate nodes that can represent their wishes, and at the same time get the voting rewards. The election of production nodes and candidate nodes is implemented through a referendum organisation. This referendum lists the options (the option to enter the poll to enter the election) for users to choose from.

5.2.4 Using a custom governance model

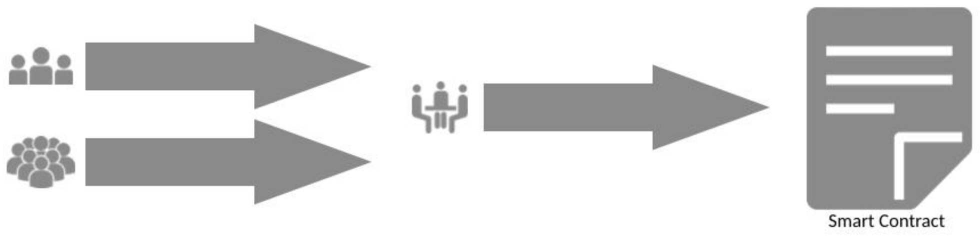

Figure: 5.2.4-1 Custom governance model (parliament organisations, referendum organisations, association organisations)

Take an example to illustrate the custom governance model in the figure above: it includes parliament organisations, referendum organisations, and association organisations. parliament organisations and referendum organisations are members of parliament organisations. In association organisations, only two members can vote for approval to trigger changes. The affairs of the parameters; in the parliamentary organisation, only when the votes in favour are more than 2/3, the votes are less than 1/5, the abstentions are less than 1/5, and the participation ratio is not less than 3/4. Yes, the organisation of the referendum has a total of no less than 1,000,000 votes, no less than 500,000 votes in favour, no more than 100,000 votes against it, and no more than 100,000 abstain from voting in the association organisation.

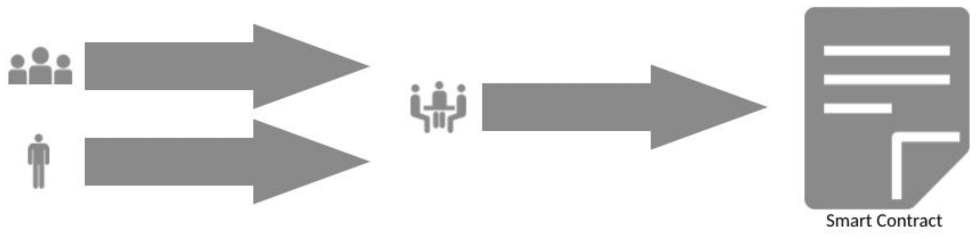

Figure: 5.2.4-2 Custom governance model (parliament organisations, developers, associations)

Take an example to illustrate the custom governance model in the figure above it consists of parliament organisations, developers, and association organisations. The difference from Figure 5.2.4-1 is that the participants of the association include the council and the developer. The affairs can be performed only if both members vote in favour.

The transactions that use a customised governance model include but are not limited:

1. Adjust users' estimated rewards for voting
   Users can get rewards by voting. According to the principle that the longer the voting lock time, the more rewards are obtained. Specific adjustable parameter range, please refer to 3.4.
2. Adjustment of important parameters in the user charging model
   Charging users involves the interests of users, developers, and production nodes, and is determined jointly by users, developers, and production nodes. This requires custom governance models based on three governance models. For specific adjustable parameter ranges, please refer to Figure 2.5.2.
3. Adjust important parameters in developer fee model
   The second chapter refers to the transaction fee model. The transaction fee model is divided into a developer payment model and a user payment model. These models are composed of deduced formulas. Among them, there are parameters for adjusting fees. The parliamentary organisation and the developer form a cooperative organisation and decide together. Specific adjustable parameter range, please refer to 2.5.1.

5.3 SideChain Governance

Three governance models also make SideChain governance easy. This section describes how to use these three basic governance models to manage SideChain transactions.

5.3.1 Using Parliament Governance Model

The affairs governed by the parliament model include but are not limited:

1. SideChain creation
   When developers want to create an exclusive SideChain, because it involves production nodes to provide computing support for this SideChain, they need to submit an application for joint resolution by the parliamentary organisation.
2. SideChain contract deployment / upgrade
   Although there is an automatic contract inspection, for security, the deployment and upgrade of the exclusive SideChain and shared SideChain contract deployment and upgrade need to be approved/deployed by the parliamentary organisation during the initial launch. (This is also the governance process of the SideChain DApp)
3. Governance of cross-chain index fee model / time-based fee model in arrears
   In the case of arrears, the cross-chain index fee model / time-based fee model is decided by the parliamentary organisation through proposal governance to determine whether it is necessary to continue to provide services for this SideChain.

5.3.2 Using Association Governance Model

The association is an autonomous organisation, and the affairs within the association can be jointly determined through association governance, such as:

1. Association members jointly manage account assets
   Consistent with the "association members jointly manage account assets" in the MainChain governance.

5.3.3 Using Customized Governance Model

The transactions governed using the custom governance model include but are not limited:

1. Adjust the parameters of the SideChain block index fee
   In order to achieve cross-chain transfers or cross-chain verification, the MainChain needs to index the blocks of the SideChain. For this SideChain, a block index fee needs to be paid. The block index fee is determined through consultation between the SideChain developer and the parliamentary organisation.
   Consistent with the transactions in Figure 5.2.4-2 in the MainChain governance.
2. Adjust important parameters in developer fee model
   Consistent with the "adjustment of important parameters in the developer fee model" transaction in the MainChain governance.
3. Adjustment of important parameters in the user charging model
   Consistent with the "adjustment of important parameters in the user fee model" in MainChain governance.

In addition, because aelf's exclusive SideChain is designed for "one chain, one scenario", developers can use a custom governance model to build a governance model that suits their scenario.