DAO Secret Telegram

Decentralized secret telegram cluster based on blockchain technology

DST DAO • 2022

Preface

In modern society, people's communications are monopolized by various communication giants. Under many established rules, people have gradually lost their freedom of communication. Traditional communications are expensive and easily eavesdropped and cracked, but vested interest groups are unwilling to upgrade technology to solve these problems. This has led to high personal communication costs, frequent privacy leaks, and people's communication security cannot be guaranteed.

DST adheres to the belief of complete decentralization and is committed to promoting the innovation of blockchain communication technology. At the same time, it upholds the concept of liberalism . To this end, DST adopts a completely open source approach and uses decentralized deployment technology. Anyone can apply to become a communication gateway on DST and provide other members with access to the DST communication gateway .

DST is a future autonomous encrypted communication ecosystem community based on web 3.0 ;

DST is the first distributed, decentralized , and self-revenue-generating free communication community ;

DST has established a social network with true freedom of communication in human society . After community members enter DST through different communication gateways, they can use private key addresses as their unique identity for communication. Through the support of technologies such as asymmetric encryption of communication messages, it solves the problems of privacy security , information leakage , and communication trust in personal communications .

At the same time, community members can also obtain the world's only DID communication number generated by DST . The communication number supports mutual calls and telephone communications between different members without

charging any communication fees , which improves the high investment disadvantages of the traditional communications industry and greatly reduces the cost of the entire communications industry, thereby truly benefiting mankind and the communications industry through blockchain encrypted communication technology.

There is no central node or hierarchical management structure in the DST network . It achieves organizational goals through interaction, competition and collaboration between network nodes from the bottom up. Therefore, the business dealings between nodes and between nodes and organizations in DST are no longer determined by administrative affiliation, but follow the principles of equality, voluntariness, reciprocity and mutual benefit, driven by each other's resource endowments, complementary advantages and win-win interests. Each organizational node will collaborate effectively under the incentive mechanism of the token based on its own resource advantages and talent qualifications, thereby generating a strong synergy effect.

The operating rules, responsibilities and rights of participants, and reward and punishment mechanisms in the code and smart contracts and DST are all open and transparent. In addition, through a series of highly efficient autonomous principles, the rights and interests of relevant participants are precisely differentiated and balanced, that is, those who work, make contributions, and assume responsibilities are matched with corresponding rights and benefits, so as to promote industrial division of labor and equal rights, responsibilities, and interests, making the organization's operation more coordinated and orderly.

Another technical feature brought by DST is the design of DVM decentralized virtual machine, which is a virtual environment that can adopt a distributed structure and run high-level languages and smart contracts. The emergence of DVM can make DST communication a new generation of Web3 infrastructure.

CONTENTS

Chapter 1 Blockchain Technology and Encrypted Social Networking

1.1 Blockchain Technology and Web3.0

Blockchain is a decentralized distributed ledger technology that has the characteristics of decentralization, transparency, and security. These characteristics make blockchain technology play an important role in Web 3.0.

In Web 3.0, blockchain is used to implement decentralized applications (DApps) and decentralized autonomous organizations (DAOs). These applications and organizations are built on blockchain technology. They are not controlled by any central agency and are entirely determined by their users and participants. This decentralized feature makes the Web 3.0 world more equal and fair, removing the control of power by central agencies and giving everyone the opportunity to participate.

In addition, blockchain technology can also be used to ensure data security and privacy. In the traditional Internet, we need to rely on central institutions to protect our data security, but this centralized approach is vulnerable to hacker attacks

and abuse. Blockchain technology can ensure data security and privacy through encryption, distributed storage and other means. This is also a very important feature of Web 3.0 . In the near future, blockchain and Web 3.0 will be widely used .

1.2 The development of the web

Web 1.0: Read-only Internet

Platform creation, platform ownership, platform control, platform benefits

In the Web 1.0 era, in order to make information sharing more convenient, the early Internet was like an information display platform, aiming to become a window for people to understand the wider world. During this period, with the development of technology, information has evolved from text to pictures and then to videos, but this information is static and read-only. Users can only be spectators and cannot participate in it.

Web 2.0: Readable and writable Internet

User-created, platform-owned, platform-controlled, platform-distributed

In the Web 2.0 era, the model will be more user-centric, allowing users to be content creators and interact and collaborate through social media. Under the model of user-created content, the scale of the Internet began to expand rapidly, but the information created essentially did not belong to the users themselves.

Web 3.0: The Internet of Value

User created, user owned, user controlled, protocol assigned

In the Web 2.0 era, the development of the Internet has deviated from its original design. As users have changed from recipients of information to participants, their digital behaviors on the Internet have been recorded one by one and controlled by various companies in the form of data. For example, Google records your search traces, Facebook and Twitter record your interpersonal relationships and interactions, Amazon records your purchase records, and so on. These technology giants gain profits through large amounts of data, and algorithms control your world.

Most importantly, your personal data does not belong to you, but has become a profit-making tool for various companies. For example, if Facebook goes bankrupt one day

and shuts down your social account, all your past updates and relationships will disappear with the disappearance of Facebook. Your information does not belong to you in essence.

The experience of Web 3.0 may not be very different from that of Web 2.0, but the difference is that users or creators can retain ownership of the content they contribute and can also receive a certain degree of rewards. In terms of privacy, users can clearly know the purpose of these data and have the right to make decisions.

With the development and accumulation of technology, the change of the times is unstoppable. Compared with the more rigid Web2.0, Web3.0, a network that can be read, written and owned, is the mainstream of the future.

1.3 The impact of blockchain and Web 3.0 on the future

Blockchain and Web 3.0 are revolutionary technologies that will have a profound impact on future society, economy, and politics.

The decentralized future

Blockchain and Web 3.0 will bring a decentralized future that will enable everyone to participate in decision-making and governance. This will bring a more equal and fair social environment, reduce the abuse and centralization of power, and increase social stability and sustainability.

A more secure and transparent network environment

Blockchain and Web 3.0 will bring a more secure and transparent network environment, which will protect user data and privacy, reduce hacker attacks and abuse. At the same time, blockchain and Web 3.0 will also improve the transparency and fairness of the network, allowing everyone to understand and join the Web3.0 network .

A more inclusive financial and economic environment

Blockchain and Web 3.0 will bring a more inclusive financial and economic environment, allowing more people to access

financial services and opportunities. At the same time, the decentralized financial system will reduce the profits and costs of financial intermediaries, making finance more fair , transparent and efficient .

Changing politics and governance

Blockchain and Web 3.0 will also change politics and governance, allowing everyone to participate in politics and governance. This will bring a more democratic and fair political environment, reduce the abuse of power and corruption. At the same time, blockchain and Web 3.0 will also improve the efficiency and transparency of governance, allowing everyone to understand and participate in governance.

1.4 The convergence of blockchain and Web 3.0 communications

Communication is the basis of human beings as a social group. Since the birth of human beings, no matter what form it takes, no matter whether it is under the control of consciousness or not, human social communication activities have never stopped. As the communication carrier of social activities, social networks have become more and more

dependent on social networks as people's social needs increase. At the same time, the development of social networks has gradually shortened the distance between people and regions. The advantages of blockchain technology have natural application advantages in the fields of anonymous communication and encrypted social networking. Blockchain technology will play a subversive role in traditional Internet. In terms of communication, it has natural advantages and common value demands when combined with it.

Blockchain technology can help communications achieve higher security. Security is crucial in the communication process. Blockchain technology can use encryption algorithms to protect the privacy and security of communication data. In addition, blockchain technology can also implement identity verification and digital signatures, allowing users to send and receive information more securely.

Blockchain technology can help communications achieve better decentralization. Traditional communication platforms are often controlled by centralized institutions, which can easily lead to information leakage and control. Blockchain technology can achieve decentralization and allow users to control their

own communication data more autonomously. For example, peer-to-peer communication can be adopted to avoid the control of centralized institutions.

Blockchain technology can help communications achieve better privacy protection. Blockchain technology can provide distributed encrypted storage and dynamic transmission links to prevent communication networks from being tracked and data eavesdropped.

1.5 The birth of DST chain communication

DST is an encrypted communication platform and digital token circulation ecosystem based on blockchain technology. It aims to provide underlying protocol support for anonymous mapping communication networks and cross-regional anonymous communications through the application of blockchain technology and the introduction of web3.0 protocol. Through a deep understanding and accumulation of the industry, as well as adherence to the belief in decentralization and liberalism, the DST chain communication platform will lead an era of anonymous networks with secure assets and full freedom.

The biggest difference between the DST public chain and the traditional blockchain operation mode is that DST uses a device interconnection routing architecture called NXN to realize a hybrid blockchain edge computing network built with mobile devices, PC devices, and server devices. The computing advantage of this network is that it can use device clusters to establish a virtual network relay, and improve the stability of Dpos nodes through a dynamic committee algorithm mechanism, achieving 100 times the performance and throughput of mainstream public chains. At the same time, the transaction fee consumption is only one ten-thousandth of the main chain. The reason is that the network built with mobile devices does not require additional investment to meet the same network security strength as the mainstream public chain. It is also an indicator that must be achieved as a public chain system with communication transmission as its goal. Based on the Dpos pledge model of the device cluster, mobile devices only need to perform necessary data signatures and communication relay network transmissions, without synchronizing complete blocks and running heavy computing loads, so it will not affect the normal use of mobile terminal

devices. This is also the basis for the DST network to achieve large-scale popularization, and it can make the DST network the world's largest edge computing equipment provider and the largest decentralized communication service cluster.

DST can establish a network sharing platform based on large-scale mobile devices, and can obtain CDN acceleration, data preloading and other services by using tokens, replacing centralized services such as Cloudflare or Bedge.

DST users can use the built-in DID function to realize Web3+Web2 aggregate verification through DID identity. At the same time, DID can minimize information verification based on zero-knowledge proof technology and provide qualification certificates to third parties without leaking privacy.

As a global mainstream communication application, DST can realize simultaneous interpretation by aggregating edge computing capabilities and ensure that the content of the conversation will not be monitored by a third party. Using DST, you can have barrier-free encrypted audio conversations with users in any country.

DST Organizational Governance

The full name of DST is Dao Secret Telegram, so DST is

designed with Dao governance as the core concept. Using a complete Dao organizational model, more like communism, the generated income will belong to the Dao organization collective, and the Dao organization managers will be elected by voting to be responsible for the distribution of collective assets, and the managers will maintain the interests of all members and promote the development of the Dao organization.

The following are the core features of DST

• DAO code governance : DST supports application voting upgrades. Through Dao, the DAO organization votes to decide whether to update, release, and apply the public chain code, which determines the development framework of DAO application governance for DST.

• Enhanced interaction : DST achieves an excellent operating experience by transforming complex RPC blockchain interaction instructions into a graphical mode. Compared with traditional blockchain interactions, users hardly perceive that they are interacting with the blockchain in real time. For example, users can click on a bubble prompt in the chat window to complete the collection of DPOS rewards. This is an

important advancement in the blockchain network operating system. It marks that traditional network users can enter the world of Web3 through DST, allowing command line operations such as POS staking, POS sharing, POS contract signing, and POS revenue extraction, which were previously only operated by professionals, to progress to a graphical interface.

• Virtualization technology : Based on DVM decentralized virtual machine technology, decentralized operation of high-level languages and smart contracts is realized. This enables many originally centralized computing services to be run in a distributed environment, such as simultaneous interpretation, video bitrate conversion, image processing and other functions. By migrating traditional centralized services to decentralization, your privacy can be effectively protected. At the same time, it is possible to run smart contract programs, including smart contracts developed by Solidity, Vyper, Bamboo, Mutan, and Serpent.

• Decentralization: Traditional communication methods rely on centralized servers to forward information, while DST communication uses a decentralized approach for encrypted transmission, with no third party involved, ensuring the security

and privacy of information while avoiding service interruptions caused by server failures or attacks .

• Openness: DST communication can protect the privacy of users. Users can choose to communicate anonymously without worrying about the leakage of personal information. This can also prevent communication records from being stored by third parties, thereby enhancing users' sense of trust.

• Security : DST Communications uses asymmetric encryption technology to ensure the security of information. Once a message is sent, it will be encrypted and only the recipient can decrypt it. This method can effectively prevent information from being stolen or monitored by hackers.

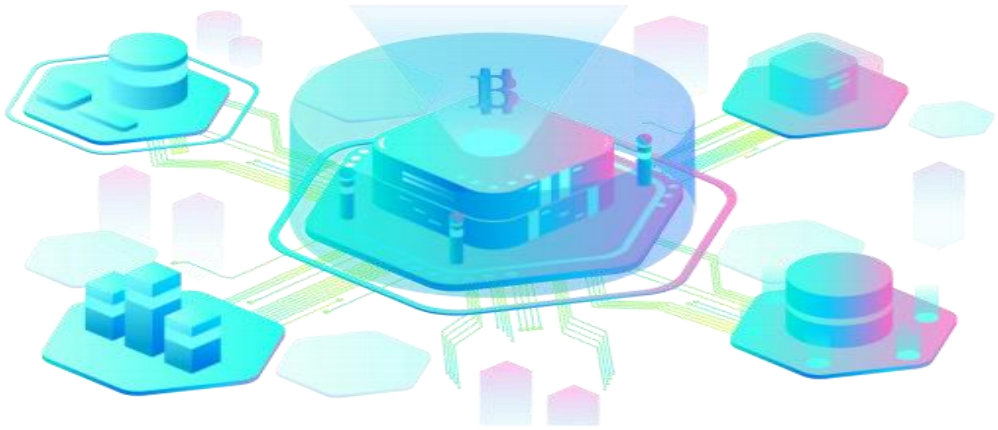
The DST blockchain communication platform is committed to building a completely anonymous and untraceable encrypted communication protocol and a token incentive model to create a universal, fully functional, high-performance, rich application scenarios, easy to use, and user-friendly decentralized, anonymous communication network and web3.0-related infrastructure.

1.6 DST Edge Computing Network

DST is not only a decentralized network communication tool, but also a distributed universal edge computing platform. By paying DST, you can host resources such as files and video streaming on the DST network, and provide network traffic acceleration based on edge computing nodes. Using SDK, you can integrate it in any Android APP and realize resource publishing and resource P2P loading capabilities, ultimately achieving data forwarding services at a cost far lower than that of IDC service providers.

Note:

- Support Android minimum SDK version: 23 or above for integration
- If you want to achieve video streaming and real-time playback, only HLS video format is supported
- It is recommended to use PC hardware to add to the edge computing platform. Larger network uplink bandwidth and computing power will bring higher weight and benefits.



Chapter 2 Dpos Consensus Mechanism

2.1 NXN Coin

The governance token is called NXN, and the total amount is capped at 21 million, of which 2.1 million will be airdropped to all users, nodes, and super nodes participating in the DST public chain test. Governance tokens are only used for voting and cannot be used as gas fees. Other tokens are obtained through POS staking mining until all are mined.

2.2 DST Coin

The platform token is called DST, with a total of 5 billion that can be used for mining destruction, gas fees, equipment cluster governance voting, and DAO organizing committee elections.

After destroying a group, members can gain chat experience points. The higher the chat level, the more gateway communication resources they can obtain.

50% of the GAS fees for transfers on the entire network will

be directly destroyed, and the DST production will be halved every 4 years.

The upgrade criteria and corresponding benefits of member chat levels are as follows:

Chat Level	Upgrade conditions	Basic usage quota bonus
1	0DST	0
2	100DST	1%
3	200DST	2%
4	400DST	3%
5	800DST	4%
6	1600DST	5%
7	3200DST	6%
8	6400DST	7%
9	12800DST	8%
10	25600DST	9%
11	51200DST	12%
12	102400DST	15%
13	204800DST	18%
14	409600DST	21%
15	819200DST	24%
16	1638400DST	27%

17	3276800DST	30%
18	6553600DST	33%
19	13107200DST	36%
20	26214400DST	39%
21	52428800DST	43%
22	104857600DST	47%
23	209715200DST	51%
24	419430400DST	55%
25	838860800DST	59%
26	1677721600DST	63%
27	3355443200DST	67%
28	6710886400DST	71%
29	13421772800DST	75%
30	26843545600DST	80%
31	53687091200DST	85%
32	107374182400DST	90%
33	214748364800DST	100%

2.3 DAO Coin

DAO coins are the fee currency for claiming computing

power mining rewards. When claiming computing power rewards, you need to pay 30% of DAO coins as a fee. DAO coins are generated when members destroy them. 10% of the destroyed amount of DAO coins are automatically mined and given to the current cluster's DAO organization pool, and at the same time, the allocation of 7.5% of the destroyed amount of DAO coins is obtained .

The production of DAO coins is halved every four years, and the collection fee is halved every four years.

2.4 NXN Dpos Market

In the blockchain network, if a device without proof of stake is added to the network, the blockchain network will be attacked or unstable, which will affect the stability and security of the entire blockchain network. The blockchain network based on the Dpos model will have two roles: device providers and token holders. Only by combining the two can a complete Dpos network be built, and both parties can also receive mining rewards.

In order to make the DST network more stable, the POS mechanism is used to reward device nodes that provide bandwidth with native tokens. Hardware device holders can

contribute their own devices and set the share ratio with token holders. This usually forms a POS pledge market, where NXN token holders and hardware holders can freely choose and compare prices to decide their partners.

However, if Dpos nodes are frequently offline, the stability of the blockchain network will be reduced, so Dpos will establish a penalty mechanism to punish device holders who are unable to ensure the operation of the device. If hardware device providers want users holding DST tokens to pledge on their own nodes, hardware device holders need to pledge a certain amount of tokens themselves to indicate that they are willing to bear the risk and cost for the stability of the hardware.

2.5 Equipment Cluster

DST holders can generate GAS computing power by destroying DST to the device cluster to conduct computing power mining and earn income.

Unlike traditional Dpos, the device cluster is equivalent to a Dpos node and is maintained by a certain number of online devices. In this way, the Dpos node realizes a distributed, highly available, loose hardware architecture. At the same time, users

of these online device clusters can also obtain a 1% commission as a reward for 30 minutes of online time every day.

The users of the device cluster are presented as communication groups. In this way, communication, DAO governance, and POS markets can be integrated, and it is also easier to invite non-professionals to become members of the DST network. Different levels are assigned to the minimum online device index, so that larger and more stable device clusters have profit advantages.

Upgrade baseline for device clusters:

Level	Overall minimum destruction	Minimum Active Node	Bridge DAO Coin Cap
1	5000	1	5000
2	10000	2	10000
3	15000	3	15000
4	20000	4	20000
5	25000	5	25000
6	30000	6	30000
7	40000	8	40000
8	50000	10	50000
9	65000	12	65000

10	80000	15	80000
11	100000	18	100000
12	120000	21	120000
13	145000	25	145000
14	175000	30	175000
15	210000	35	210000
16	255000	41	255000
17	310000	48	310000
18	375000	57	375000
19	455000	67	455000
20	550000	79	550000
21	660000	93	660000
22	795000	109	795000
23	955000	128	955000
24	1150000	150	1150000
25	1385000	175	1385000
26	1665000	204	1665000
27	2000000	238	2000000
28	2405000	278	2405000
29	2890000	325	2890000
30	3470000	379	3470000

31	4165000	442	4165000
32	5000000	515	5000000
33	6000000	600	6000000

2.6 DVM Virtual Machine

The public chain network that conventionally supports smart contracts uses a smart contract virtual machine to carry all smart contract operations and calculates the GAS redemption ratio according to a consistent standard.

This method is designed to be simple and easy to understand, but it cannot be sharded and support higher loads, so the cost of running smart contracts on existing public chains is high, especially for high-concurrency smart contract projects.

DST solves the problems of contract execution concurrency and communication data sharding by establishing a DVM virtual machine module.

DVM is implemented through built-in Layer2 virtualization isolation sharding technology, and as a key feature for realizing CDPOS collaborative device clusters,

DVM technology allows CDPOS nodes to run smart

contracts on their own and broadcast smart contract operation results. When running contracts through DVM, it can provide computing power for contracts specified by DVM creators to reduce the fees consumed when running at the L1 layer. DVM

will share hardware resources with DST general virtual machines. DST-generated virtual machines are destroyed by coin holders and hosted in a certain device cluster. The destruction volume determines the proportion of DVM's operating hardware resources, the daily upper limit of computing power provided for contracts, and DPOS revenue.

DST introduces GAS computing power value to achieve dynamic conversion between DST tokens and DVM computing power weight, which can maintain the development of the virtual machine ecosystem at any stage.

2.7 GAS computing power

GAS computing power is a unit of measurement used to represent the GAS fees that can be deducted from each DVM virtual machine every day . The daily upper limit of the deduction is dynamically calculated by an intelligent algorithm. GAS computing power will not decrease with the deduction and will last forever. Having more GAS computing power can

execute smart contracts for a longer period of time without paying additional fees .

Holders of GAS computing power packaged into private DVM virtual machines can participate in DST's full-network DPOS mining. The full-network DPOS income is 360,000 DST per day, which is weighted according to the proportion of private DVM virtual machine GAS computing power in the total network.

2.8 DAO Organization Governance Pool

DAO organization governance pool, each device cluster registered on the chain has a DAO organization governance pool for the cluster, which is the fund storage center for the decentralized management of the cluster and a DVM virtual machine owned by the entire cluster. The use and rental of cluster DVM virtual machines will be decided by DAO organization voting and corresponding to DPOS rewards, which will enter the DAO organization governance fund pool and be allocated according to the settings.

To ensure the stable development of the equipment cluster, the cluster will determine the cluster level based on the number

of online devices, the total DVM computing power and the online rate, and open DAO governance capabilities based on the level.

Whenever someone in the cluster is destroyed, the DAO governance pool will generate an additional 5% DST and 10% DAO coin reward for the DAO organization governance pool, which will be distributed according to the ratio set by the cluster. The DST in the DAO organization governance pool will be claimed and used by voting, and the DAO coins will be automatically allocated in order according to the destruction queue.

The DST DAO organization allocation ratio and the cluster DVM DAO organization allocation ratio are 51% at the lowest and 100% at the highest. The group owner can enter the settings and can only modify it once a day. The DAO coin DAO organization allocation ratio is 0% at the lowest and 100% at the highest. The group owner can enter the settings and can only modify it once a day.

2.9 DST Incentives and Outputs

By destroying DST in the device cluster and exchanging it for computing power, and packaging it into a DVM virtual

machine , you can get Dpos incentives. The incentive ratio is obtained according to the inflation formula. DST mining will be enabled 3 days after the mainnet goes online .

The basic formula for DST's daily output is:

$$360{,}000\ coins\ are\ produced\ every\ 14{,}400\ blocks$$

DST exchange GAS computing power:

$$\text{Real-time exchange ratio} = \frac{360{,}000}{\text{Total GAS computing power of the entire network}} \times 100$$

GAS mining power:

$$\text{GAS computing power mining} = \frac{360000}{\text{Total cluster GAS computing power}} \times \text{Total GAS computing power held}$$

1% of the DST produced by DVM every day is given to the cluster's equipment as a commission reward to encourage the equipment to stay online. 99% of DPOS rewards are owned by users. The mined DST will be automatically pledged to the SWAP POS pool for mining. When redeeming DST, 30% of the DAO tokens are required, and various tokens mined in the SWAP POS pool contract will be obtained at the same time.

2.10 Cluster Routing Bridge Rewards

At the same time, in order to encourage different device clusters to expand to form a multi-node distributed network

and strengthen the communication flow between clusters, a cluster bridging incentive algorithm is established. The incentive is issued to the cluster Dao governance pool in the form of computing power and DAO coins. The number of routing bridge layers is one layer. Each activity of the routing bridge device cluster will have a probability of generating DAO coin rewards, but this reward is closely related to the online rate of the device cluster. At the same time, the total amount of this reward will not exceed 70% of the Dao reward of the bridge device cluster.

bridge incentive algorithm for cluster computing power :

Bridge cluster additional computing power* Online rate* 5 %*random number

bridge incentive algorithm of the cluster DAO coin :

New DAO coin rewards for bridge cluster * 70% *online rate of bridge cluster*random number

The bridge reward computing power is given to the upper-level DAO organization DVM, and the DAO coins are distributed to the cluster owner and the DAO organization governance pool according to the DAO coin distribution ratio set internally by the upper-level cluster. It is triggered when the

cluster has active operations, and the DAO coins are distributed in sequence according to the cluster's internal destruction queue.

Active on-chain operations of the cluster include: claiming equipment rewards, redeeming DST, group owners claiming wages, destroying DST to obtain DVM computing power, adding and kicking people to the cluster, DVM computing power authorization, and updating cluster administrators.

The upper limit of DAO coins obtained varies with cluster levels. When the upper limit of DAO coins obtained by a cluster is reached , the generated DAO coins are accumulated in the bridge pool and will be distributed after the cluster is upgraded .

When the online rate of the cluster is greater than 60% , settlement is made at 100% . When it is less than 60% , deductions are made based on the online ratio .

2.11 NXN Incentives and Outputs

NXN will only be distributed in the form of airdrops in the early stage, and the airdrop targets include active test users, user nodes, and super nodes. Staking NXN can obtain mining incentives between 3% and 7% per year. As the staking ratio

increases, the inflation coefficient will show a linear decline.

NXN is the governance token of the DST chain and the proof-of-work token of the gateway DPOS node. The DST chain adopts the DPOS mechanism. The block generation and verification in the DST network are completed by the POS gateway nodes. The system will automatically select the top 51 nodes with the largest number of mortgages and the most stable online as POS nodes (hereinafter referred to as gateway nodes). Gateway nodes are dynamic. If the gateway node is disconnected and the link cannot be connected, other qualified backup nodes will automatically replace it. Gateway nodes are dynamic to ensure that the network will never be destroyed.

In order to ensure the stable operation of the main chain network, DPOS nodes are given a daily mining reward of 36,000 DST, and weighted mining is carried out according to the NXN staked by each node. The main responsibility of the mining node is to verify transactions on the blockchain and generate new blocks. This is a feature of the consensus algorithm based on DPoS, which requires verification nodes to compete to become block generators to generate new blocks to maintain the normal operation and block generation of the network.

2.12 DAO Coin Incentives and Output

DAO coins are only generated when there is new destruction in the cluster. The output is 10% of the destruction amount. It is distributed to the group owner and the DAO organization pool according to the ratio set in the cluster. The DAO organization pool is triggered when the cluster has an on-chain operation, and DAO coins are distributed in order according to the destruction queue in the cluster.

2.13 Halving Redemption Contract

Whenever DST is halved for four years, the DST staked in the SWAP POS pool can manually choose to trigger the redemption contract. After the trigger, 10% will enter the 4-year halving redemption contract and be released in 333 days. The released DST does not require the matching 30% DAO coins can be collected directly.

Chapter 3 DST Chain Supporting Service Components

3.1 DST Communication System

Based on the main chain, DST chain can build one-to-one encrypted video instant messaging within the platform. Its most prominent features are decentralization, high-strength encryption, anti-traffic tracking, support for edge computing and digital identity DID. Through content-oriented, it attracts users to actively participate in the platform, such as social chat, video conference, live interaction, creating topics, etc.

In one-on-one video chats, users can speak freely without being influenced by the platform. Based on the peer-to-peer communication method, the decentralized network structure has an unchangeable characteristic. When applied to social platforms, it can re-establish the credit system between people.

At the same time, DST Chain's unique encryption technology ensures the security of one-to-one video instant messaging data transmission. Encryption and privacy protection technology are the core foundation of DST Chain. DST Chain can encrypt user privacy information that needs to

be kept confidential, ensuring that the information is only transmitted or shared between specific users.

DST uses ECC elliptical curve encryption technology to encrypt your communication data with the best performance and strength. ECC is an asymmetric encryption algorithm mode and has been widely used in various data encryption scenarios.

Both parties use an asymmetric encryption algorithm to generate a pair of public and private keys , and send their own public keys to each other. In the communication process, they will use the other party's public key to encrypt a randomly generated data, and then send the encrypted data to the other party . The receiving party will use their own private key to decrypt the data . Therefore, this method can provide higher security, because even if hackers steal the public key, they cannot decrypt the communication content, and everyone's communication information on the entire network is unique.

3.2 DST Wallet Payment System

DST wallet system is powerful and is mainly developed based on the technical infrastructure of mainstream wallets. It has also passed CertiK's security audit certification . The wallet

design pursues the ultimate user experience, making the payment environment safer and asset management more convenient. It supports multiple currencies and multiple languages to create safe and easy-to-use digital asset management tools and entry-level applications. At the same time, it is simple and easy to operate, has been tested for tens of thousands of times, and has powerful anti-theft technology to maximize the security of users' digital assets.

Decentralization: Blockchain technology means that the DST wallet payment system does not require a centralized institution (such as a bank), but rather the nodes on the network jointly maintain and verify transactions, which can reduce risks and costs in the payment process and improve security and transparency.

Security: The security of DST wallet is based on cryptography technology. Each transaction is strictly encrypted and verified to ensure the security of the transaction. In addition, due to the decentralized nature of the blockchain, no one can destroy the entire payment system by attacking a single central institution.

Transparency: The transaction information of the DST chain is publicly recorded on the blockchain and can be viewed by

anyone. This transparency can promote trust and reduce the risk of fraud.

Speed and efficiency: Compared with traditional bank transfers, blockchain payments can complete transactions faster and more efficiently because blockchain does not require intermediaries to verify transactions.

Cost saving: DST payment reduces the involvement of intermediaries and reduces transaction costs. At the same time, since the transaction is decentralized, some additional fees and handling fees are avoided.

3.3 D vm virtual machine module

D vm module makes the DST chain highly scalable and fully compatible and interoperable with the Ethereum Virtual Machine (EVM). It is built on the CometBFT (a branch of Tendermint Core) consensus engine to achieve fast finality, high transaction throughput, and short block time .

Support multiple languages: D VM can run smart contracts in multiple programming languages, including Solidity, Vyper, Lity, Bamboo, etc. This allows developers to use the programming language they are best at to write smart

contracts for the DST chain .

Higher security: D VM provides a series of built-in security features, such as code review, data isolation, and permission management, which can help developers write more secure smart contracts.

Faster execution speed: D VM can improve the execution speed of smart contracts through various technical means, such as code optimization, caching mechanism, etc., which enables smart contracts to respond to user requests faster.

Higher interoperability: D VM is based on the Ethereum virtual machine and can interoperate with other applications and smart contracts in the Ethereum ecosystem, which makes it easier for developers to integrate their applications with the DST chain .

3.4 DAPP Market

DST The Dapp application market is a global, peer-to-peer distributed application distribution system, whose goal is to promote the rapid development and implementation of DST chain applications. The principle is based on distributed storage and peer-to-peer network node transmission, which can make

DAPP downloads faster, safer, more robust, and more durable.

DAPP is the abbreviation of Decentralized Application, which is called distributed application/decentralized application in Chinese. Generally speaking, different DAPPs will adopt different underlying blockchain development platforms and consensus mechanisms, or issue tokens on their own (they can also use universal tokens based on the same blockchain platform).

DAPP's different underlying blockchain development platforms are like the iOS and Android systems of mobile phones, which are the underlying ecological environment of each DAPP. DAPP is a variety of distributed applications derived from the underlying blockchain platform ecology, and is also a basic service provider in the blockchain world. DAPP is to blockchain what APP is to IOS and Android.

Through the side chain, applications developed based on Ethereum can pass DST The Dapp application market implements a DAPP precise indexing system based on content search, a distributed application distribution system based on peer-to-peer network transmission technology, a data-based bounty system, a traffic-based mining system, and a

download-based repurchase and destruction system.

3.5 DAO Governance

The DST network is managed in the form of Dao (Decentralized Autonomous Organization), that is, the entire blockchain network can be updated by voting, advocating that everyone has the right to own and control their own digital identity, which can safely store their personal data and protect privacy. In the DST framework, individuals can have a digital identity that contains all this information and is managed separately, rather than having multiple centralized providers manage multiple digital identities. Management is coded, programmed, and automated. "Code is law", organizations are no longer pyramidal but distributed, power is no longer centralized but decentralized, management is no longer bureaucratic but community autonomy, and organizational operations no longer require companies but are replaced by highly autonomous communities. In addition, because DAO (Decentralized Autonomous Organization) operates under an operating standard and collaboration model jointly determined by stakeholders, consensus and trust within the organization

are easier to achieve, which can minimize the organization's trust costs, communication costs, and transaction costs.

3.6 DST Number (DID Identity)

The DST number is a distributed, open, and scalable naming system based on the DST chain. It is the digital identity DID of web 3.0. In the DST network, users can obtain a completely anonymous blockchain network identification number. The DST number can be generated off the network. You can use this number to store your digital assets in various blockchain networks, including but not limited to Eth, BNB, Magic, and the DST network. Through the DST node network, each number can easily communicate with each other through text, voice, images, files, etc.

The job of a DST number is to map human-readable numbers (such as "888888888") to machine-readable identifiers (such as DST addresses, other cryptocurrency addresses, content hashes, and metadata). FNS also supports "reverse resolution" to associate metadata such as a canonical name or interface description with a DST address.

Top-level numbers, such as "88888XXXX" and "99999XXXX", are held by DPOS validators called communication gateways,

which specify the rules that govern the allocation of their numbers. Anyone can obtain ownership of a DID for their own use, following the rules specified in the contracts of these gateways.

The DST number is an identity certificate in the future decentralized society, owned and controlled by individuals, while solving other identity needs such as the confirmation, verification, storage, management and use of personal information.

3.7 SWAP

SWAP Decentralized Exchange (DEX) is a trading platform that operates on the DST blockchain network. Its operation mode is different from that of traditional centralized exchanges. In decentralized exchanges, transactions are not conducted through central institutions, but through smart contracts on the DST blockchain network. Compared with traditional exchanges, it has the following advantages:

Decentralized architecture: DEX has no central server or centralized exchange architecture. Transactions are conducted directly on the blockchain, using smart contracts to execute and

verify transactions.

User-held private keys: Users retain their private keys in decentralized exchanges, which means they have full control over their assets. Funds do not need to be stored in a centralized exchange wallet.

Smart Contract Execution: Transactions are executed through smart contracts, which are automated, intermediary-free code. Smart contracts are responsible for managing transaction processes such as order matching and fund clearing, ensuring the transparency and credibility of transactions.

Asset Exchange: In a decentralized exchange, users can exchange digital assets directly without going through a third party. This asset exchange can include cryptocurrencies, tokens, etc.

Global and borderless: Because they are based on blockchain technology, decentralized exchanges are usually global, and users are not restricted by geographical location. Any user with an Internet connection can participate.

Anonymity and Privacy: Some decentralized exchanges offer users a higher degree of anonymity because users do not need

to provide personally identifiable information to conduct transactions.

Security: Since funds are kept in the user's wallet rather than in the account of a centralized exchange, DEX is generally more difficult to be targeted by hackers. In addition, the automatic execution of smart contracts reduces potential human errors.

Chapter 4 Technical Support

DST Mainchain SDK Introduction

This SDK includes modules such as auth, bank, and capability. Developers can use the SDK to develop their own digital currency wallets, blockchain browsers, cross-chain bridges, and other ecological applications.

4.1 Auth

Application transaction and account authentication

The auth module is responsible for specifying basic transactions and account types for the application, as the SDK itself is agnostic to these details. It contains the AnteHandler, which performs all basic transaction validity checks (signatures,

nonces, auxiliary fields, etc.), and exposes the account keeper to allow other modules to read, write, and modify accounts.

(1) Concept

gas & fees

For network operators, fees have two purposes:

Fees limit the growth of the state stored by each full node and allow general censorship of transactions of little economic

value. Fees work best as an anti-spam mechanism in situations where validators have no interest in the use and identity of the network.

Fees are determined by the gas limit and gas price, where

fees = ceil(gasLimit * gasPrices)

Txs incur gas costs for all state reads/writes, signature verification, and fees proportional to the size of the transaction. Operators should set a minimum gas price when starting their node. The cost per unit of gas for each token denomination you wish to support must be set:

simd start ... --minimum-gas-prices=0.00001stake;0.05photinos

When adding a transaction or idle transaction to the memory pool, the validator checks whether the gas price of the transaction (determined by the provided fees) meets any of the validator's minimum gas prices. In other words, a transaction

must provide a fee of at least one denomination that matches the validator's minimum gas price.

Tendermint does not currently offer fee-based memory pool priority, fee-based memory pool filtering is node-local and not part of consensus. But with the setting of a minimum gas price, such a mechanism could be implemented by node operators.

Because the market value of tokens fluctuates, validators are expected to dynamically adjust their minimum gas price to a level that incentivizes use of the network. Because the market value of tokens fluctuates, validators are expected to dynamically adjust their minimum gas price to a level that incentivizes use of the network.

(2) Status

Account

An account contains authentication information for an external user that is uniquely identifiable on the SDK blockchain, including a public key, address, and account number/serial number for replay protection. For efficiency, since the account balance must also be captured to pay fees, the account

structure also stores the user's balance as `sdk.Coins'.

Module clients wishing to add more account types can do so by exposing the account as an interface and storing it internally as a base account or a member account.

0x01 | Address -> ProtocolBuffer (account)

Account Interface

The account interface exposes methods for reading and writing standard account information. Note that all of these methods operate on the account structure of the confirmation interface - in order to write the account to storage, you need to use account keeper.

AccountI is an interface for storing currency at a given address within a state. It assumes a notion of sequence numbers for replay protection, a notion of account numbers for replay protection against previously modified accounts,

and a public key for authentication.

```
type AccountI interface {
    proto.Message
```

GetAddress() sdk.AccAddress

SetAddress(sdk.AccAddress) error // errors if already set.

GetPubKey() crypto.PubKey // can return nil.

SetPubKey(crypto.PubKey) error

GetAccountNumber() uint64

SetAccountNumber(uint64) error

GetSequence() uint64

SetSequence(uint64) error

// Ensure that account implements stringer

String() string

Basic Account

A Basic Account is the simplest and most common type of account, it just stores all the necessary fields directly in a structure.

BaseAccount defines a base account type. It contains all the necessary fields for basic account functionality. Any

custom account type should extend this type to implement additional functionality (e.g. vesting).

```
message BaseAccount {
    string address = 1;
    google.protobuf.Any pub_key = 2;
    uint64 account_number = 3;
    uint64 sequence = 4;
}
```

Vesting Account

Vesting Account will be introduced in detail later.

AnteHandlers

Handlers

The auth module currently has no transaction handlers of its own, but exposes a special `AnteHandler' that does basic validity checking on a transaction so it can be tossed out of the memory pool. Note that the ante handler is called on `CheckTx', but also on `DeliverTx', since Tendermint originators currently have the ability to include transactions that fail `CheckTx' in the blocks they originate.

Ante Handler

anteHandler(ak AccountKeeper, fck

```
FeeCollectionKeeper,  tx sdk.Tx)
        if !tx.(StdTx)
            fail with "not a StdTx"
        if isCheckTx and tx.Fee < config.SubjectiveMinimumFee
            fail with "insufficient fee for mempool inclusion"
        if tx.ValidateBasic() != nil
            fail with "tx failed ValidateBasic"
        if tx.Fee > 0
            account = GetAccount(tx.GetSigners()[0])
            coins := acount.GetCoins()
            if coins < tx.Fee
                fail with "insufficient fee to pay for transaction"
        account.SetCoins(coins - tx.Fee)
        fck.AddCollectedFees(tx.Fee)
        for index, signature in tx.GetSignatures()
        account = GetAccount(tx.GetSigners()[index])
        bytesToSign          :=          StdSignBytes(chainID,
acc.GetAccountNumber(),
        acc.GetSequence(), tx.Fee, tx.Msgs, tx.Memo)
        if !signature.Verify(bytesToSign)
            fail with "invalid signature"
```

return

Keepers

Keepers

The auth module exposes only one keeper, the account keeper, which can be used to read and write accounts.

Account Keeper

There is currently only one fully authorized account manager, which has the ability to read and write all fields of all accounts, and iterate over all stored accounts.

// AccountKeeperI is the interface contract that x/auth's keeper implements.

type AccountKeeperI interface {

// Return a new account with the next account number and the specified address. Does not save the new account to the store.

NewAccountWithAddress(sdk.Context, sdk.AccAddress) types.AccountI

// Return a new account with the next account number. Does not save the new account to the store.

NewAccount(sdk.Context, types.AccountI) types.AccountI

// Retrieve an account from the store.

GetAccount(sdk.Context, sdk.AccAddress) types.AccountI

// Set an account in the store.

SetAccount(sdk.Context, types.AccountI)

// Remove an account from the store.

RemoveAccount(sdk.Context, types.AccountI)

// Iterate over all accounts, calling the provided function. Stop iteration when it returns false.

IterateAccounts(sdk.Context, func(types.AccountI) bool)

// Fetch the public key of an account at a specified address

GetPubKey(sdk.Context, sdk.AccAddress)

(crypto.PubKey, error)

// Fetch the sequence of an account at a specified address.

GetSequence(sdk.Context, sdk.AccAddress) (uint64, error)

// Fetch the next account number, and increment the internal counter.

GetNextAccountNumber(sdk.Context) uint64

}

vesting

Examples

Simple example

Given a continuous attribution account with 10 attribution coins.

OV = 10

DF = 0

DV = 0

BC = 10

V = 10

V' = 0

Receive 1 coin immediately

BC = 11

Time has passed, 2 coins vest

V = 8

V' = 2

Delegate 4 coins to Verifier A

DV = 4

BC = 7

Give out 3 coins

BC = 4

More time passed, and 2 more coins were delivered

V = 6

V' = 4

2 coins were sent. At this point, the account cannot send any

more until more coins are attributed to it or more coins are received. However, it can still be delegated.

BC = 2

Advanced Examples

Same initial conditions as in the simple example.

As time goes by, the ownership of the 5 coins

V = 5

V' = 5

Delegate 5 coins to Validator A

DV = 5

BC = 5

Delegate 5 coins to Verifier B

DF = 5

BC = 0

Validator A is slashed by 50%, making the value of the tokens delegated to A now 2.5 coins.

Revoke authorization from Validator A (2.5 coins).

DF = 5 - 2.5 = 2.5

BC = 0 + 2.5 = 2.5

Revoke the delegation (5 coins) from Validator B. The account can now only send 2.5 coins unless it receives more coins or until more coins are attributed. However, it can still delegate.

dv = 5 - 2.5 = 2.5

df = 2.5 - 2.5 = 0

bc = 2.5 + 5 = 7.5

Please note that we have an excess of DV.

Parameters

| Key| Type | Example|

|------------------------------------| ----------

----------|----------- |

| MaxMemoCharacters | uint64 | 256 |

| TxSigLimit | uint64 | 7 |

| TxSizeCostPerByte | uint64 | 10 |

| SigVerifyCostED25519 | uint64 | 590 |

| SigVerifyCostSecp256k1 | uint64 | 1000 |

4. 2 Authz

x/authz is an implementation of an SDK module that allows arbitrary permissions to be granted from one account (grantor) to another account (grantee). Authorization must be granted one by one for a specific Msg service method using an implementation of the authorization interface.

(1) Concept

Authorization

Any concrete authorization type defined in the x/authz module must satisfy the authorization interface outlined below. Authorizations determine exactly which permissions are granted. They are extensible and can be defined for any Msg service method, even outside the module that defines the Msg method. Authorizations use the new ServiceMsg type of ADR 031.

Built-in authorization

The authz module comes with the following authorization types:

Send authorization

SendAuthorization implements the authorization interface for bank.v1beta1.Msg/Send ServiceMsg, it receives a SpendLimit and updates it to zero.

GenericAuthorization

GenericAuthorization implements the Authorization interface, which grants unrestricted permission to execute the provided ServiceMsg on behalf of the grantor's account.

Gas

To prevent DoS attacks, granting StakeAuthorizaitons with x/authz will incur gas. StakeAuthorizaitons allows you to authorize another account to delegate, undelegate, or re-delegate a validator. Authorizers can define a list of validators they will allow and/or deny delegation to. The SDK will iterate over these lists and charge 10 gas for each validator in both lists.

(2) Status

A grant is identified by the combination of the grantor

address (the byte address of the grantor), the grantee address (the byte address of the grantee), and the ServiceMsg type (its method name).

AuthorizationGrant:0x01|granter_address_len(1byte)|granter_address_bytes|grantee_address_len(1byte)|grantee_address_bytes|msgType_bytes->ProtocolBuffer(AuthorizationGrant)

(3) Message

Next, the message processing of the authz module is described.

Message/Approval Authorization

Create a grant authorization using the MsgGrantAuthorization message.

This message is expected to fail in the following cases.

1. Both the grantor and the grantee have the same address.

2. Provide an expiration time that is less than the current Unix timestamp.

3. The authorization provided was not fulfilled.

Message/Revoke Authorization

The allowed authorization can be removed via the MsgRevokeAuthorization message.

This message is expected to fail in the following cases.

1. Both the grantor and the grantee have the same address.
2. The provided MethodName is empty.

MsgExecAuthorizedRequest

When a grantee wants to execute a transaction on behalf of a grantor, it must send MsgExecAuthorizedRequest.

This message is expected to fail in the following situations.

1. No authorization is granted for the information provided.
2. The authorized person does not have the authority to run transactions.
3. The approved authorization has expired.

(4) Events

The authz module emits the following events.

keeper

Grant Authorization:

Type	Attribute Key	Attribute Value
grant-authorization	module	authz
grant-authorization	grant-type	{msgType}
grant-authorization	granter	{granterAddress}
grant-authorization	grantee	{granteeAddress}

Revoke authorization

Type	Attribute Key	Attribute Value
revoke-authorization	module	authz

revoke-authorization	grant-type	{msgType}
revoke-authorization	granter {granterAddress}	
revoke-authorization	grantee {granteeAddress}	

4. 3 Bank

The bank module is responsible for handling multi-asset coin transfers between accounts, and for tracking special-case pseudo-transfers that must work differently with specific types of accounts (particularly delegated/non-delegated attribution accounts) . It exposes several interfaces with different capabilities for secure interaction with other modules that must change a user's balance.

Additionally, the banking module tracks and provides support for querying the total supply of all assets used in the application.

(1) Status

The x/bank module maintains the state of three main objects, account balances, denom metadata, and the

total supply of all balances.

Supply: 0x0 | byte(denom) -> byte(imount)

denom metadata : 0x1 | byte(denom) -> ProtocolBuffer(Metadata)

Balance: 0x2 | byte(address length) | []byte(address) | []byte(balance.Denom) ->ProtocolBuffer(balance)

(2) Keeper

The bank module provides a number of exported keeper interfaces that can be passed to other modules that read or update account balances. Modules should use the least-privileged interface that provides the functionality they need.

(3) Message

Message sending

handleMsgSend(msg MsgSend)

inputSum = 0

for input in inputs

```
        inputSum += input.Amount
    outputSum = 0
    for output in outputs
        outputSum += output.Amount
    if inputSum != outputSum:
        fail with "input/output amount mismatch"
    return inputOutputCoins(msg.Inputs, msg.Outputs)
```

(4) Events

The bank module emits the following events

MsgSend, MsgMultiSend

In addition to handling program events, the bank keeper generates events when the following methods are called (or any method that ultimately calls these methods):

MintCoins, BurnCoins, addCoins, subUnlockedCoins/DelegateCoins

(5) Parameters

The bank module contains the following parameters

Key	Type	Example

SendEnabled []SendEnabled [{denom: "stake", enabled: true }]

DefaultSendEnabled bool true

4. 4 Capability

(1) Concept

ability

Capabilities are multi-owner. A domain keeper can create a capability via `NewCapability`, which creates a new unique, unforgeable object - the capability reference. Newly created capabilities are automatically persisted; calling modules do not need to call `ClaimCapability`. Calling `NewCapability` will create the capability with the calling module and name, as a tuple, which is considered the first owner of the capability.

Capabilities can be claimed by other modules, which will take them as owners. `ClaimCapability' allows a module to claim

a capability key it received from another module, so that future `GetCapability' calls will succeed. If a module that receives a capability wishes to access it by name in the future, it must call `ClaimCapability'. Likewise, capabilities are

multi-owner, so if multiple modules have a reference to a capability, they will all own it. If a module receives a capability from another module but does not call `ClaimCapability', it can use it within the execution of the transaction, but will not be able to access it afterwards.

Any module can call "AuthenticateCapability" to check whether a capability actually corresponds to a particular name (which can be untrusted user input) that was associated with the module before calling it.

`GetCapability' allows a module to get a capability that it has previously claimed by name. The module is not allowed to retrieve capabilities that it does not possess.

(2) Status

Index

CapabilityOwners

Capability

4. 5 Crisis

The crisis module stops the blockchain in case a

blockchain invariant is broken. The invariant can be registered with the application during the application initialization process.

(1) Status

Fixed costs

Since the expected gas cost to verify a variable is high (and potentially exceeds the maximum allowed block gas limit), a constant fee is used instead of the standard gas consumption method. The constant fee is intended to be larger than the expected gas cost of running the constant with the standard gas consumption method. The ConstantFee parameter is saved in the global params storage.

(2) Message

MsgVerifyInvariant

The invariance of the blockchain can be checked using the `MsgVerifyInvariant` message.

The message is expected to fail if .

- The sender does not have enough coins to pay the fixed fee

- Immutable route is not registered

The message checks the invariance provided, and if the invariance is broken, it will pannic, stopping the blockchain from functioning. If the invariance is broken, the fixed fee will never be deducted, as the transaction was never submitted to the block (effectively being refunded). However, if the invariance is not broken, the fixed fee will not be refunded.

(3) Events

The Crisis module contains the following events

Handlers

MsgVerifyInvariance

Type	Attribute Key	Attribute Value
invariant	route	{invariantRoute}
message	module	crisis
message	action	verify_invariant
message	sender	{senderAddress}

(4) Parameters

The Crisis module contains the following parameters

| Key| Type | Example|

|-------------------|-------------------------------------
----------- -|---------------------------------|
| ConstantFee | object (coin) |
{"denom":"uatom","amount":"1000"} |

4. 6 Distribution

This simple allocation mechanism describes a functional way to passively distribute rewards between validators and delegators. Please note that this mechanism does not allocate funds as precisely as the active reward allocation mechanism, so it will be upgraded in the future.

The mechanism works as follows. The collected rewards are pooled globally and passively distributed to validators and delegators. Each validator has the opportunity to charge a fee to the delegator for the rewards collected on behalf of the delegator. The fees are collected directly into the global reward pool and the validator proposer reward pool. Due to the nature of passive accounting, whenever the parameters that affect the reward distribution

rate change, the extraction of rewards must also occur.

Whenever a withdrawal is made, the maximum amount they are entitled to must be withdrawn, without leaving anything in the pool. Whenever tokens are bound, unbound, or re-delegated to an existing account, a full withdrawal of rewards must occur (because the rules of lazy accounting change). Whenever a validator chooses to change the commission of rewards, all accumulated commission rewards must be withdrawn at the same time.

The allocation mechanism outlined here is used to lazily distribute the following rewards between validators and associated delegators.

1. Atomic regulation of socially allocated multi-token fee proposer reward pool inflation

2. Commission fees on all rewards earned by validators for their delegators' stake are aggregated in a global pool, as well as validator-specific proposer reward pools. The mechanism used allows validators and delegators to withdraw their rewards independently and lazily.

(1) Concept

In a Proof of Stake (PoS) blockchain, rewards earned from transaction fees are paid to validators. The fee distribution module distributes the rewards fairly to the constituent delegators of the validator.

Rewards are calculated on a period basis. Whenever a validator's delegation changes, for example, when a validator receives a new delegation, the period is updated. The rewards for a single validator can then be calculated by taking the total rewards for the period before the delegation started and subtracting the current total rewards.

A commission is paid to a validator when the validator is removed or when the validator requests a withdrawal. The commission is calculated and incremented at each BeginBlock operation to update the accumulated fee amount.

Rewards to delegators are distributed when a delegator is changed or removed, or when a withdrawal is requested. All slashes to validators that occurred during the current

delegation period are applied before rewards are distributed.

(2) Status

FeePool

All global tracking allocation parameters are stored in the FeePool. Rewards are collected and added to the reward pool, and from here they are distributed to validators/authorizers.

Note that the rewards pool holds decimal coins (`DecCoins`) to allow for fractional coins to be earned from operations such as inflation. When coins are distributed from the rewards pool, they are truncated back to `sdk.Coins`, which are non-decimal.

FeePool: 0x00 -> ProtocolBuffer(FeePool)

```
// coins with decimal
type DecCoins []DecCoin
    type DecCoin struct {
    Amount sdk.Dec
    Denom string
```

}

Validator Distribution

The certifier assignment data of the relevant certifiers will be updated every time.

The amount delegated to a validator is updated, the validator successfully proposes a block and gets rewarded, any delegator quits from the validator, or the validator withdraws its commission.

ValidatorDistInfo:0x02|ValOperatorAddrLen(1byte)|ValOperatorAddr->ProtocolBuffer (validatorDistribution)

type ValidatorDistInfo struct {

OperatorAddress sdk.AccAddress

SelfBondRewards sdk.DecCoins

ValidatorCommission types.ValidatorAccumulatedCommission

}

Delegation Distribution

Each delegation distribution only needs to record the height of its last extraction fee. Since a delegation must extract

fees (aka bonded tokens, etc.) when its attributes change, its attributes will remain unchanged, and the delegation's _accumulation_ coefficient can be calculated dynamically, just knowing the height of the last extraction and its current attributes.

DelegationDistInfo: 0x02 | DelegatorAddrLen (1 byte) | DelegatorAddr | ValOperatorAddrLen (1 byte) | ValOperatorAddr -> ProtocolBuffer(delegatorDist)

type DelegationDistInfo struct {

WithdrawalHeight int64 // last time this delegation withdrew rewards

}

(3) Begin Block

In every `BeginBlock`, all fees received from the previous block are transferred to the allocated `ModuleAccount` account. When a delegator or validator withdraws their rewards, they withdraw from the `ModuleAccount`. During the BeginBlock, different claims on the fees collected are updated as follows.

- The previous block proposer and his/her delegates receive a fee reward ranging from 1% to 5%.

- Reserve community tax is collected.

- The remaining portion is distributed to all bound validators in proportion to their voting rights

To incentivize validators to wait and include additional pre-commits in blocks, the block proposer's reward is calculated from the Tendermint pre-commit messages.

(4) Message

Set withdrawal address (MsgSetWithdrawAddress)

By default, the withdrawal address is the delegator's address. To change its withdrawal address, the delegator must send a `MsgSetWithdrawAddress' message.

Changing the withdrawal address is only possible if the parameter `WithdrawAddrEnabled` is set to `true`.

The withdrawal address cannot be an account in any module. These accounts are added to the allocation holder's `blockedAddrs' array during initialization, thereby preventing them from becoming withdrawal addresses.

func (k Keeper) SetWithdrawAddr(ctx sdk.Context, delegatorAddr sdk.AccAddress, withdrawAddr sdk.AccAddress) error

if k.blockedAddrs[withdrawAddr.String()] {

fail with "`{withdrawAddr}` is not allowed to receive external funds"

}

if !k.GetWithdrawAddrEnabled(ctx) {

fail with `ErrSetWithdrawAddrDisabled`

}

k.SetDelegatorWithdrawAddr(ctx, delegatorAddr, withdrawAddr)

(5)Hooks

The available hooks that can be called by this module, and the hooks that are called from this module.

Create or modify authorization assignments

triggered-by: staking.MsgDelegate, staking.MsgBeginRedelegate, staking.MsgUndelegate

Before

- Delegation rewards are withdrawn to the delegator's withdrawal address. Rewards include the current period, excluding the starting period.

- Validator period has been increased. Because the power and stake distribution of validators may have changed, the validator period has been increased.

- The reference count of the principal's starting period is decremented.

After

The delegator's starting height is set to the previous period. Due to the existence of the before-hook, this period is the last period for the delegator to receive rewards.

(6) Events

BeginBlocker

| Type | Attribute Key | Attribute Value |

|------------------------|------------------|--

-------------------------|

| proposer_reward | validator | {validatorAddress} |

| proposer_reward | reward | {proposerReward} |

| commission | amount | {commissionAmount} |

| commission | validator | {validatorAddress} |

| rewards | amount | {rewardAmount} |

| rewards | validator | {validatorAddress} |

(7) Parameters

| Key| Type | Example|

| ------------------ |------ ------------ | --------

------------------ |

| communitytax | string (dec)| "0.020000000000000000" [0] |

| baseproposerreward| string (dec)

|"0.010000000000000000"[0]

|

bonusproposerreward|string(dec)|"0.040000000000000000"[0]

| withdrawaddrenabled | bool | true |

4. 7 Evidence

Any specific type of evidence submitted to the 'evidence' module must fulfill the 'Evidence' contract outlined below . Not all specific types of evidence fulfill this contract in the same way, and some data may be completely irrelevant for certain types of evidence. An additional 'ValidatorEvidence', which extends 'Evidence', has also been created to define an evidence contract that is designed to protect against malicious validators.

Governancetype Evidence interface {

```
proto.Message
    Route() string
        Type() string
String() string
Hash() tmbytes.HexBytes
ValidateBasic() error
    // Height at which the infraction occurred
GetHeight() int64
}

type ValidatorEvidence interface {
Evidence

// The consensus address of the malicious validator at
time of infraction
GetConsensusAddress() sdk.ConsAddress

// The total power of the malicious validator at time of
infraction
GetValidatorPower() int64
```

// The total validator set power at time of infraction

GetTotalPower() int64

}

(1) Status

Currently, the x/evidence module only stores valid submitted evidence in the state. The evidence state is also stored in the GenesisState of the x/evidence module and exported.

GenesisState defines the initial state of the evidence module.

message GenesisState {

// evidence defines all the evidence at genesis.

repeated google.protobuf.Any evidence = 1;

}

(2) Message

Evidence is submitted via the MsgSubmitEvidence message.

// MsgSubmitEvidence represents a message that supports submitting arbitrary

// Evidence of misbehavior such as equivocation or counterfactual signing.

message MsgSubmitEvidence {

string submitter = 1;

google.protobuf.Any evidence = 2;

}

event

MsgSubmitEvidence

| Type | Attribute Key | Attribute Value |

|---------- | ---------- | ---------- |

| submit_evidence | evidence_hash | {evidenceHash} |

| message | module | evidence |

| message | sender | {senderAddress} |

| message | action | submit_evidence |

(3) Parameters

This module does not contain any parameters

4. 8 Mint

(1) Concept

Mining Mechanism

The purpose of the mining mechanism is

Allowing a flexible inflation rate, determined by market demand, targets a specific bond-to-equity ratio that strikes a

balance between market liquidity and stock supply. The moving change rate mechanism ensures that if the collateral percentage exceeds or falls below the target collateral percentage, the inflation rate will adjust to further incentivize or disincentivize collateralization. Setting the target %bonding rate to less than 100% encourages the network to keep some unbonded tokens, which should help provide some liquidity.

The moving rate of change mechanism ensures that if the collateralization percentage exceeds or falls below the target collateralization percentage, the inflation rate will adjust to further incentivize or disincentivize collateralization.

If the inflation rate is below the target % binding rate, the inflation rate will increase until it reaches a maximum value.

If the target bond rate of 67% is maintained, then the inflation rate will remain unchanged.

If the inflation rate is above the target bonding rate, the inflation rate will decrease until it reaches a minimum value.

(2) Status

Miner

A miner is a space that stores information about the current inflation.

Minter: 0x00 -> ProtocolBuffer (minter)

Parameters

Mining parameters are saved in global parameters.

(3) Initial Block

The minting parameters are recalculated at the start of each block and pay for inflation.

(4) Parameters

| Key| Type | Example|

|--------------------|----------------|---------

------------------------------|

| MintDenom | string | "uatom" |

| InflationRateChange | string (dec) | "0.130000000000000000" |

| InflationMax | string (dec) | "0.200000000000000000" |

| InflationMin | string (dec) | "0.070000000000000000" |

| GoalBonded | string (dec) | "0.670000000000000000" |

| BlocksPerYear | string (uint64) | "6311520" |

(5) Events

Beginblocker

| Type | Attribute Key | Attribute Value |

|--------|------------------------------|------- ------
----------------|

| mint | bonded_ratio | {bondedRatio} |

| mint | inflation | {inflation} |

| mint | annual_provisions | {annualProvisions} |

| mint | amount | {amount} |

4. 9 Params

The params package provides a globally available parameter storage.

There are two main types, Keeper and Subspace. A subspace is an independent namespace used for parameter storage where keys are prefixed with a pre-configured space name. Keepers have a permission to access all existing spaces.

Subspace can be used by a single keeper who needs a private parameter storage space that other keepers cannot

modify. The params Keeper can be used to add routes to the `x/gov` router in order to modify any parameters when a proposal is passed.

(1) Keeper

During application initialization, Keeper.Subspace can be used to allocate subspaces for keepers of other modules and

store them in Keeper.space. These modules can then obtain references to their specific parameter storage via Keeper.GetSubspace.

(2)Subspace

Subspace is a subspace with a prefix of "subspace" where parameters are stored. Each module that uses parameter storage will take a subspace to isolate access permissions.

4. 10 Slashing

(1) Status

Signature information (Liveness)

Each block includes a set of pre-commitments made by validators of the previous block, which is called the LastCommitInfo provided by Tendermint. As long as the LastCommitInfo contains pre-commitments of +2/3 of the total voting power, it is valid.

Proposers are incentivized to include pre-commitments from all validators in their Tendermint LastCommitInfo by charging an additional fee proportional to the difference between the voting power included in LastCommitInfo and +2/3 (see Fee Distribution).

type LastCommitInfo struct {

Round int32

Votes []VoteInfo

}

Validators are penalized for failing to include a certain number of blocks in LastCommitInfo, are automatically imprisoned, potentially deleted, and are unbound.

(2) Message

Unbind

If an authenticator is automatically unjailed due to downtime, and wishes to come back online and possibly rejoin the bound set, it MUST send MsgUnjail.

// MsgUnjail is a sdk.Msg used to unjail a bound authenticator, returning it to

// They are returned to the set of bound validators so they can start receiving regulations

// Rewards.

message MsgUnjail {

string validator_addr = 1;

}

If the validator has enough stake to enter the top n = MaximumBondedValidators, it will automatically be re-bound, and all delegators still delegating to the validator will be re-bound and begin collecting provisions and rewards again.

(3) Initial Block Effectiveness Tracking

At the start of each block, we update each validator's ValidatorSigningInfo and check if they have crossed the activity threshold within a sliding window. This sliding window is defined by the SignedBlocksWindow and the index of this window is determined by the IndexOffset in the validator's ValidatorSigningInfo. For each block processed, whether the validator signed or not, the IndexOffset is incremented. Once the index is determined, the MissedBlocksBitArray and MissedBlocksCounter are updated accordingly.

Finally, to determine if the validator has crossed the validity threshold, we get the maximum number of missed blocks, maxMissed, which is SignedBlocksWindow - (MinSignedPerWindow * SignedBlocksWindow) and the

minimum height we can determine validity, minHeight. If the current block is larger than minHeight, and the validator's MissedBlocksCounter is greater than maxMissed, they will be deleted by SlashFractionDowntime, will be jailed for DowntimeJailDuration, and the following values will be reset: MissedBlocksBitArray, MissedBlocksCounter, and IndexOffset.

(4)Hooks

Cutting hook

The slashing module implements the StakingHooks defined in x/staking to record validator information. These hooks should be registered in the staking module's structure during the application's initialization process.

The following hooks affect the reduction status.

AfterValidatorBonded creates a ValidatorSigningInfo instance as described below.

AfterValidatorCreated stores a validator's consensus key.

AfterValidatorRemoved removes a validator's consensus key.

Bound Authenticator

After successfully binding a new validator for the first time, we create a new ValidatorSigningInfo structure for the now bound validator with the StartHeight of the current block.

onValidatorBonded(address sdk.ValAddress)

```
signingInfo, found = GetValidatorSigningInfo(address)
if !found {
signingInfo = ValidatorSigningInfo {
StartHeight : CurrentHeight,
IndexOffset : 0,
JailedUntil : time.Unix(0, 0),
Tombstone : false,
MissedBloskCounter : 0
}
setValidatorSigningInfo(signingInfo)
```

}

return

(5) Events

The slashing module emits the following event/tag

MsgServer

MsgUnjail

| Type | Attribute Key | Attribute Value |

| ---------- | ---------- | ---------- -- |

| message | module | slashing |

| message | sender | {validatorAddress} |

Keeper

BeginBlocker: HandleValidatorSignature

| Type | Attribute Key | Attribute Value |

| --- | --- | --- |

| slash | address | {validatorConsensusAddress} |

| slash | power | {validatorPower} |

| slash | reason | {slashReason} |

| slash | jailed [0] | {validatorConsensusAddress} |

[0] Only included if the validator is jailed.

| Type | Attribute Key | Attribute Value |

| ------------ | ------------------ | ----------------------- |

| liveness | address | {validatorConsensusAddress} |

| liveness | missed_blocks | {missedBlocksCounter} |

| liveness | height | {blockHeight} |

Jail

| Type | Attribute Key | Attribute Value |

| -------- | ------------------ | ------------------------ |

| slash | jailed | {validatorAddress} |

(6) Reduction monument

In the current implementation of the "slashing" module, when the consensus engine notifies the state machine of a validator's consensus error, the validator is partially slashed and

placed in a "jail period", a period of time during which it is not allowed to rejoin the validator set. However, due to the nature of consensus errors and ABCI, there can be a delay between when a violation occurs and when evidence of the violation reaches the state machine (this is one of the main reasons why the unbinding period exists).

(7) Parameters

| Key| Type | Example|

| ---------- | ---------- | ---------- - |

| SignedBlocksWindow | string (int64) | "100" |

| MinSignedPerWindow | string (dec) | "0.500000000000000000" |

| DowntimeJailDuration | string (ns) | "600000000000" |

| SlashFractionDoubleSign | string (dec) | "0.050000000000000000" |

| SlashFractionDowntime | string (dec) | "0.010000000000000000" |

4. 11 Staking

This module is capable of supporting advanced

proof-of-stake systems where holders of the chain's native stake tokens can become validators and can delegate tokens to validators, ultimately determining the system's effective validator set.

(1) Status

State

astTotalPower tracks the total amount of bound tokens recorded during the last EndBlock. Store entries prefixed with "Last" must remain unchanged until EndBlock.

LastTotalPower: 0x12 -> ProtocolBuffer (sdk.Int)

Params

Params is a module-wide configuration structure that stores system parameters and defines the overall functionality of the Stakeout module.

Validator

A validator can have one of three statuses:

1. Unbound: Validators are not in the active set. They cannot sign blocks and cannot earn rewards. They can host delegations.

2. "Bound": Once a validator receives enough bound tokens, they are automatically added to the active set during the EndBlock and their status is updated to "Bound". They are signing and earning rewards. They can receive further delegations. They can be slashed due to misconduct. Validators who unbind their delegators must wait for the duration of UnbondingTime (a chain-specific parameter), during which they can still get the source validator slashed if their illegal behavior occurred during the token binding period.

3. Unbonding: When a validator leaves the active set by choice, or due to slashing, sawing, or tombstoning, all of their delegations will begin to unbond. All delegations must then wait the UnbondingTime before their tokens can be moved from the BondedPool to their accounts.

Delegation

Delegators are identified by combining DelegatorAddr (the address of the delegator) with ValidatorAddr. Delegators are indexed in the storage as follows:

Delegation: 0x31 | DelegatorAddrLen (1 byte) | DelegatorAddr | ValidatorAddrLen (1 byte) | ValidatorAddr ->

ProtocolBuffer(delegation)

Delegator Shares

When a token is allocated to a validator, they are allocated a certain number of delegators' shares based on a dynamic exchange rate, which is calculated based on the total number of tokens delegated to the validator and the number of shares issued so far:

Shares per Token = validator.TotalShares() / validator.Tokens()

Only the number of shares received is stored in the "Delegation Entry". When the delegator then cancels the delegation, the number of tokens they received is calculated based on the number of shares they currently hold and the reverse e.

(2)State Transitions

State transitions in the validator are executed on each EndBlock to check for changes in the active ValidatorSet.

Unbound to Bound

When a validator's rank in the ValidatorPowerIndex exceeds

the LastValidator, the following transition occurs.

Set validator.Status to Bonded

Transfer authenticator token from NotBondedTokens to BondedPool ModuleAccount

Delete existing records from ValidatorByPowerIndex

Add new update records to ValidatorByPowerIndex

Updates this validator's Validator object

Delete any ValidatorQueue record for this validator if it exists

Bound to Unbound

Updates this validator's Validator object

Set validator.Status to Unbonded

(3) Message

Msg/CreateValidator

Create a validator using the Msg/CreateValidator service message. The validator must be created using an operator's initial delegation.

Msg/EditValidator

Commission rates can be updated using the Msg / EditCandidacy service message.

Msg/Delegate

In this service message, the delegator provides coins and in return a certain number of validators (newly created) delegator shares allocated to the delegation are assigned to the delegator.

Msg/Undelegate

The Msg/Undelegate service message allows a delegator to revoke its token from the authentication process.

Msg/BeginRedelegate

The reauthorization command allows the delegator to switch validators instantly. Once the binding period is lifted, the reauthorization will be done automatically in EndBlocker.

(4) Initial Block

Each time the sequential start block is called, historical information is stored and pruned according to the

HistoricalEntries parameter.

Historical information tracking

If the HistoricalEntries parameter is 0, BeginBlock does nothing.

Otherwise, the latest historical information will be stored under the key historyInfoKey | height, and all entries older than height-HistoricalEntries will be deleted. In most cases, this will result in a single entry being pruned per block. However, if the parameter HistoricalEntries has been changed to a lower value, then there will be multiple entries in the store that must be pruned.

(5) Final Block

Each abci exit block call updates the queue and validator set operations specifying the changes to be performed.

Validator Set Changes

During this process, the set of staked validators is updated through state transitions that are run at the end of each block. As part of this process, all updated validators are also passed back to Tendermint to be included in the Tendermint validator

set, which is responsible for validating Tendermint messages at the consensus layer.

Generation

In equity, some state transitions are not instantaneous, but occur over a period of time (usually no-stickiness). When these transitions mature, some actions must be performed to complete the state operation. This is achieved by using a queue that is checked/processed at the end of each block.

Unbind Authenticator

When a validator is kicked out of the bound validator set (either by imprisonment or by not having enough bound tokens), it will begin the unbinding process and all of its delegations will also begin unbinding (while still being delegated to that validator). At this point the validator is called an "unbound validator" and will mature to become an "unbound validator" after a period of unbinding.

The validator queue is checked for mature unbonded validators at each block (i.e., completion time <= current time,

completion height <= current block height). At this point, any mature validators that do not have any remaining delegations are removed from the state . For all other mature unbonded validators that still have remaining delegations, validator.Status switches from type.Unbonding to types.Unbonded.

Unbinding Delegation

Complete the unbinding of all entries in the UnbondingDelegations.UnbondingDelegations queue with the following procedure:

Transfer the balance of coins to the representative's wallet address

Delete the complete entry from UnbondingDelegation.Entries

If there are no entries left, delete the UnbondingDelegation object from storage.

Reauthorization

Unbinding all mature reauthorizations is done through

the following process. Entries in the reauthorization queue:

Remove mature entries from Redelegation.Entries

If there are no entries left, the reauthorization object is deleted from storage.

(6)Hooks

Other modules can register actions to be executed when specific events occur in the pile. These events can be registered to be executed before or after the pile event (depending on the hook name). The following hooks can be registered in the pile:

①AfterValidatorCreated(Context, ValAddress)

Called when creating a validator

②BeforeValidatorModified(Context, ValAddress)

Called when the authenticator's status changes

③AfterValidatorRemoved (Context , ConsAddress , ValAddress)

Called when a validator is deleted

④AfterValidatorBonded (Context , ConsAddress , ValAddress)

Called when binding an authenticator

⑤AfterValidatorBeginUnbonding(Context, ConsAddress, ValAddress)

Called when the authentication process starts to unbind

⑥BeforeDelegationCreated(Context, AccAddress, ValAddress)

Called when creating a delegate

⑦BeforeDelegationSharesModified(Context, AccAddress, ValAddress)

Called when modifying the entrusted share

⑧BeforeDelegationRemoved (Context , AccAddress , ValAddress)

Called when deleting a delegate

(7) Events

This module exposes the following events

EndBlocker

Service Messages

Msg/CreateValidator

Msg/EditValidator

Msg/Delegate

Msg/Undelegate

Msg/BeginRedelegate

(8) Parameters

Key| Type	Example
-----|----------------------|
UnbondingTime	string (time ns)	"259200000000000"
MaxValidators	uint16	100
KeyMaxEntries	uint16	7
HistoricalEntries	uint16	3
BondDenom	string	"stake"
PowerReduction	string	"1000000"