DAO Secret Telegram

Decentralized secret telegram cluster based on blockchain technology

DST DAO • 2022

Preface

In modern society, people's communications are monopolized by communications giants. Under many established rules, people gradually lose their freedom of communication. Moreover, traditional communications are expensive and easy to be eavesdropped and cracked. However, interest groups are unwilling to upgrade technology to solve these problems. This makes personal communication costs remain high, privacy is frequently leaked, 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 while adhering to 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 ecological community based on web 3.0 ;

DST is the first distributed and decentralized free communication community with its own revenue-generating capabilities ;

DST has built a real social network with free communication in human society . After community members enter DST through different communication gateway entrances , they can use the private key address as a unique identity to communicate, and are supported by asymmetric encryption of communication information and other technologies to solve the problem . It solves the privacy and security of personal communications , information leakage , communication trust and other issues .

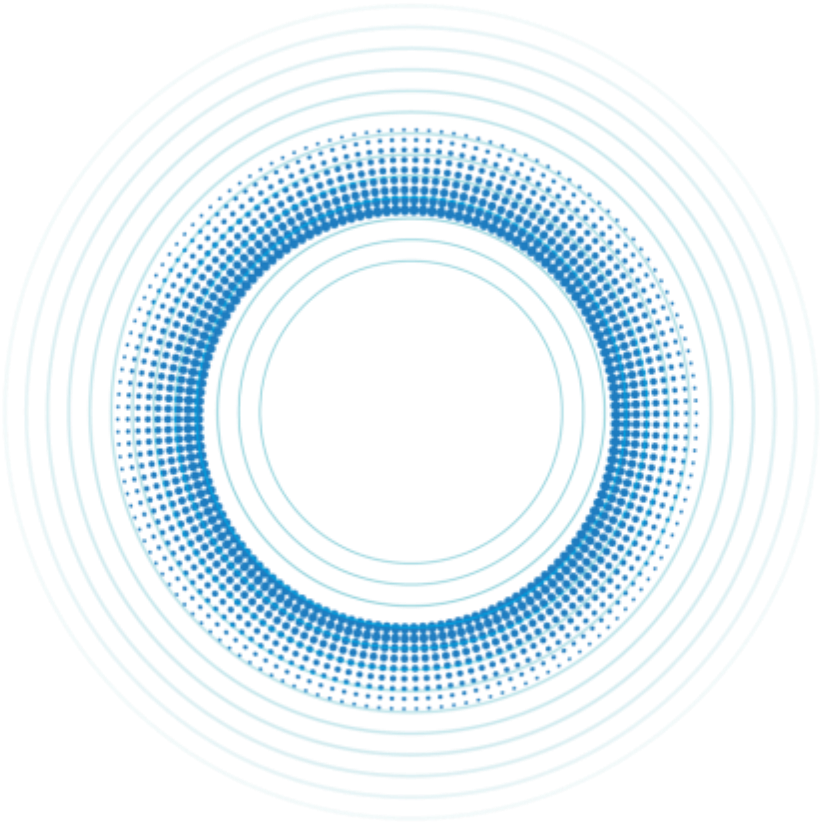
At the same time, community members can also obtain the world's only DID communication number generated by DST . The communication number supports

mutual dialing and telephone communication between different members, and does not charge any communication fees , which improves the shortcomings of high investment in the traditional communication industry and enables The cost of the entire communication industry has been significantly reduced, thus truly benefiting mankind and the communication industry through blockchain encrypted communication technology.

There are no central nodes and hierarchical management structures in the DST network . It achieves organizational goals through bottom-up interaction, competition and collaboration between network nodes. Therefore, business transactions between nodes and between nodes and organizations in DST are no longer determined by administrative affiliations, but follow the principles of equality, voluntariness, reciprocity and mutual benefit, and are determined by each other's resource endowments, complementary advantages and interests . Driven by win-win. Each organizational node will collaborate effectively under the incentive mechanism of the token based on its own resource advantages and talents, thus creating a strong synergistic effect.

Relying on the operating rules of code and smart contracts and DST , the responsibilities and rights of participants, as well as reward and punishment mechanisms, etc. are all open and transparent. In addition, through a series of efficient autonomy principles, the rights and interests of relevant participants are accurately differentiated and balanced, that is, those individuals who work, contribute, and assume responsibilities are matched with corresponding rights and benefits to promote industrial division of labor and rights, responsibilities , Equality of interests makes the organization's operation more coordinated and orderly.

Another technical feature brought by DST is the design of the DVM decentralized virtual machine, which is a virtualized 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, security, etc. 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 (DAO). These applications and organizations are built on blockchain technology. They are not controlled by any central authority and are entirely determined by their users and participants. This decentralized feature makes the world of Web 3.0 more equal and just, removing the control of power by central agencies and giving everyone the opportunity to participate.

Additionally, blockchain technology can be used to ensure the security and privacy of data. In the traditional Internet, we need to rely on central institutions to protect our data security, but this centralized approach is prone to hackers 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 history of web Internet

Web 1.0: The Read-Only Internet

Platform creation, platform ownership, platform control, platform benefit

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

Web 2.0: A readable and writable Internet

User creation, platform ownership, platform control, platform distribution

In the Web 2.0 era, its model will be more user-centered, allowing users to act as content creators and interact and collaborate through social media. Under the model of user-created content, the scale of the Internet has begun to expand rapidly, but the nature of the information created does not belong to the user himself.

Web 3.0: 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 track. As users change from recipients of information to participants, their digital behaviors on the Internet are also recorded one by one and mastered 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, etc. These technology giants make profits from large amounts of data, and algorithms control your

world.

The most important thing is that your personal data does not belong to you, but has become a profit tool for various companies. For example, if Facebook goes bankrupt one day and closes your social account, all the updates you posted in the past and all the relationships you established will also disappear with the disappearance of Facebook. Your information does not essentially belong to you. .

The experience of Web 3.0 may not be too different from that of Web 2.0, but the difference is that users or creators can retain ownership of the content they contribute and receive a certain degree of return. In terms of privacy, users can clearly know the purpose of these data and have decision-making power.

With the development and accumulation of technology, the changes of the times are unstoppable. Compared with the seriously solidified Web2.0, the "readable + writable + ownable" Internet of Web3.0 will be the mainstream in the future.

1.3 The impact of blockchain and Web 3.0 on the future

The transformative technologies of blockchain and Web 3.0 will have a profound impact on future society, economy, and politics.

The future of decentralization

Blockchain and Web 3.0 will bring about a decentralized future that will allow everyone to participate in decision-making and governance. This will bring about a more equal and just social environment, reduce the abuse and concentration 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 users' data and privacy and 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 and make 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 lead to a more democratic and fair political environment and 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 Integration of blockchain and Web 3.0 communications

Communication is the foundation of human beings as a social group. Since the birth of human beings, regardless of the form, whether under the control of consciousness or not, human social communication activities have never stopped. As a communication carrier for social activities, social networks are increasingly 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 between people and regions. The advantages of blockchain technology have natural application advantages in the fields of anonymous communication and encrypted social interaction. Blockchain technology will play a subversive role in the traditional Internet. As far as communication is concerned, its combination has natural advantages and common value demands.

Blockchain technology can help communications achieve greater security. During communication, security is crucial. 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 organizations, which can easily lead to information leakage and control. Blockchain technology can achieve decentralization and allow users to control their communication data more autonomously. For example, point-to-point communication can be used to avoid the control of centralized organizations.

Blockchain technology can help communications achieve better privacy protection. Blockchain technology can provide distributed encrypted storage and dynamic transmission links to avoid communication network tracking and data eavesdropping.

1.5 The birth of DST chain communication

DST is an encrypted communication platform and digital token circulation ecology based on blockchain technology. It aims to provide underlying protocol support for anonymous

mapping communication networks and cross-regional anonymous communication through the application of blockchain technology and the introduction of the web3.0 protocol. Through profound understanding and accumulation of the industry, as well as persistence in decentralization belief and liberalism, the DST chain communication platform will lead an anonymous network era with asset security and full freedom.

The biggest difference between the DST public chain and the traditional blockchain operation method is that DST uses a device interconnection routing architecture called NXN to implement a hybrid blockchain edge computing network built with mobile devices, PC devices, and server devices. The computing advantage of this kind of network is that it can use device clusters to establish virtual network relays, and improves the stability of Dpos nodes based on the dynamic committee algorithm mechanism, achieving 100 times the performance and throughput compared to mainstream public chains, while also reducing handling fee consumption. One ten thousandth of the main chain. The reason is that a network built using mobile devices can meet the same network security intensity as mainstream public chains without investing more costs. At the

same time, it is also a public chain targeting communication transmission. Indicators that the system must reach. Based on the Dpos staking model of device clusters, mobile devices only need to perform necessary data signatures and communication relay network transmissions. There is no need to synchronize complete blocks and run 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 enable the DST network to become the world's largest edge computing equipment provider and the largest decentralized communication service cluster.

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

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

As a global mainstream communication application, DST can realize simultaneous interpretation functions by aggregating edge computing capabilities and ensuring that the conversation content will not be monitored by third parties. 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, which is more like communism, the benefits generated will belong to the Dao organization collective, and the Dao organization managers will be responsible for allocating collective assets through voting, and the managers will safeguard the interests of all members and promote Dao Organizational development.

The following core features of DST

• DAO code governance : DST supports application voting upgrades. The Dao organization votes to decide whether to update, release and apply the public chain code, which determines the development structure 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 block interaction, users have almost no perception that they are interacting with the blockchain in real time. Interaction, for example, the user can click on a bubble prompt mark in the chat window to complete the collection of Dpos rewards. This is an important advancement for blockchain network operating systems. It marks that traditional Internet users can enter the world of Web3 through DST, allowing the command line operations of POS staking, POS sharing, POS signing, and POS revenue extraction that only professionals can operate in the past to a graphical interface.

• Virtualization technology: Based on DVM decentralized virtual machine technology, it realizes the decentralized operation of high-level languages and smart contracts, which can run many originally centralized computing services in a distributed environment, such as simultaneous interpretation, video Functions such as code rate conversion and image processing can effectively protect your privacy by migrating traditional centralized services to decentralization. At the same

time, you can 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 method for encrypted transmission without the participation of a third party, ensuring the security and privacy of information while avoiding the risk of Service interruption due to server failure or attack .

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

• Security : DST communication uses asymmetric encryption technology to ensure the security of information. Once a message is sent, it is 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 strives to create a completely anonymous and untraceable encrypted

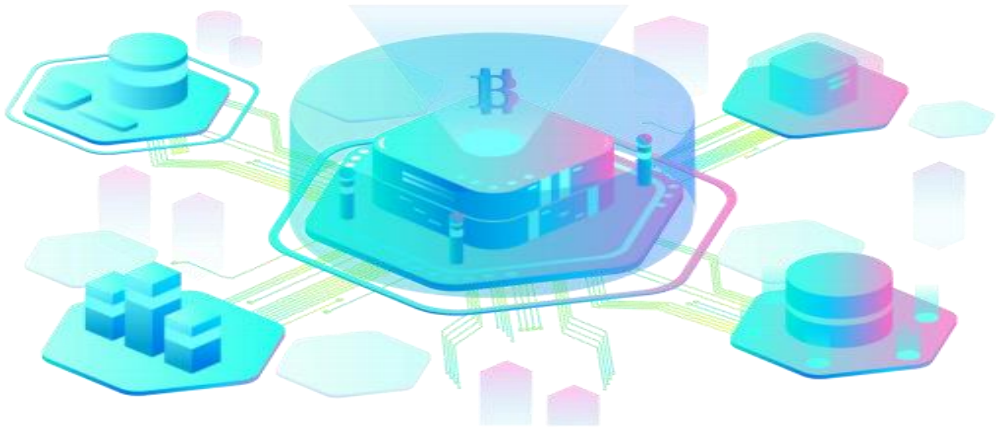
communication protocol and Token incentive model to create a decentralized system that is universal, has complete support functions, high performance, rich application scenarios, easy to use, and good user experience. , anonymous communication network and infrastructure related to web3.0.

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, resources such as files and video streaming media can be hosted on the DST network, and network traffic can be accelerated based on edge computing nodes. , using the SDK, you can integrate it in any Android APP and realize resource publishing and P2P resource loading capabilities, ultimately realizing data forwarding services that are far lower than the cost of IDC service providers.

Precautions:

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



Chapter 2 Dpos Consensus Mechanism

2.1 Governance Token

The name of the governance token is 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 is mined.

2.2 Platform Token

The name of the platform token is DST, which can be used for pledge mining, Gas fees, equipment cluster governance voting, and Dao organization committee elections.

2.3 Dpos market

In the blockchain network, if devices without proof of rights are added to the network, the blockchain network will be attacked or unstable, which will affect the stability and security of the entire blockchain network. A blockchain network based on the Dpos model will have two roles: equipment provider and token holder. Only by combining the two can a complete Dpos network be constructed, and both parties can also obtain mining rewards.

In order to make the DST network more stable, a POS mechanism is used to reward device nodes that provide bandwidth with native tokens. Hardware device holders can contribute their own devices and set a share ratio with token holders. This usually forms a POS pledge market, where token holders and hardware holders can freely choose and compare prices to determine their partners.

However, if Dpos nodes frequently go offline, it will reduce the stability of the blockchain network. Therefore, Dpos will establish a penalty mechanism to punish equipment holders who are unable to ensure the operation of the equipment. Hardware equipment providers who wish to hold DST tokens

Users pledge on their own nodes, and hardware device holders need to pledge a certain amount of tokens themselves to show that they are willing to bear risks and costs for hardware stability.

2.4 Device cluster

DST holders can generate income by staking DST to the device cluster.

Different from 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 implements a distributed and highly available loose hardware architecture.

Users of device clusters are displayed 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.

Equipment clusters can be assigned different levels based on the total pledged amount and the minimum online equipment indicator, which gives larger and more stable equipment clusters a revenue advantage.

Upgrade baseline for device clusters:

Grade	Overall minimum amount burned	lowest active node
1	5000	1
2	10000	2
3	15000	3
4	20000	4
5	25000	5
6	30000	6
7	40000	8
8	50000	10
9	65000	12
10	80000	15
11	100000	18
12	120000	21
13	145000	25
14	175000	30
15	210000	35
16	255000	41
17	310000	48
18	375000	57

19	455000	67
20	550000	79
21	660000	93
22	795000	109
23	955000	128
24	1150000	150
25	1385000	175
26	1665000	204
27	2000000	238
28	2405000	278
29	2890000	325
30	3470000	379
31	4165000	442
32	5000000	515
33	6000000	600

2.5 DVM virtual machine

A public chain network that conventionally supports smart contracts, the smart contract virtual machine is used to host all smart contract operations and calculate GAS prices and fees according to consistent standards.

This method should be simple in design and easy to understand but cannot be fragmented and support higher loads. Therefore, the running cost of 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 fragmentation by establishing a DVM virtual machine module.

DVM is implemented through built-in Layer2 virtualization isolation sharding technology, and is a key feature for realizing CDPOS collaborative device clusters. Through DVM technology, CDPOS nodes can run smart

contracts by themselves, broadcast smart contract operation results, and run the contract through DVM. , can provide computing power to the contract specified by the DVM creator to reduce the handling fees consumed when running at the L1 layer.

DVM will share hardware resources with the DST general virtual machine. The currency holder will destroy Dst to generate a virtual machine and host it in a certain equipment cluster. The amount of destruction determines the proportion of DVM running hardware resources and the daily computing

power provided for the contract. upper limit, and DPOS income.

By introducing GAS computing power value, DST realizes the dynamic conversion between DST tokens and DVM computing power weight, and can maintain the development of the virtual machine ecology at any stage.

2.6 GAS computing power

GAS computing power is a unit of measurement used to represent the daily deductible GAS fee for each DVM virtual machine . The daily deduction limit is 1% of the GAS computing power. The GAS computing power will not be reduced with the deduction and will Lasts forever, with more GAS computing power you can execute smart contracts for a longer period of time without paying extra fees .

Holding GAS computing power and packaging it as a private DVM virtual machine, you can participate in DST's network-wide DPOS mining. The network-wide DPOS income is 360,000 DST per day, which is weighted and distributed according to the total GAS computing power of the private DVM virtual machine in the entire network.

2.7 DAO Organizational Governance Pool

DAO organization governance pool. Each equipment cluster registered on the chain has the DAO organization governance pool of the cluster. It is the fund storage center for decentralized management of the cluster. It is the DVM virtual machine owned by the cluster and the cluster DVM virtual machine. The use and leasing will be decided through voting by the DAO organization, and the corresponding DPOS rewards will enter the DAO organization governance fund pool and be distributed according to the settings.

In order to ensure the stable development of the equipment cluster, the cluster will determine the cluster level based on the number of online devices and the total amount of DVM computing power, and open DAO governance capabilities based on the level.

2.8 DST incentives and outputs

By staking DST to the equipment cluster, you can obtain Dpos incentives, and the incentive ratio is obtained according to the inflation formula.

The basic formula for daily output of DST is:

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

DST exchange for GAS computing power:

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

GAS computing power mining quantity:

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

At the same time, in order to encourage the expansion of different equipment clusters to form a multi-node distributed network and strengthen the communication flow between clusters, a cluster connectivity incentive algorithm was established. The incentives are distributed to the cluster Dao governance pool in the form of DST, and are determined by the elected committee. Members make proposals and allocations.

Cluster connectivity incentive algorithm:

GAS computing power defined at the cluster level* Online rate* 5%* (communication traffic weight)

Cluster DAO governance incentive algorithm:

The current cluster has added GAS computing power * 10%

2.9 NXN incentives and output

In the early stage, NXN will only be distributed in the form of airdrops. The airdrop targets include active test users, user nodes, and super nodes. By staking NXN, you can obtain mining incentives ranging from 3% to 7% per year. As the staking ratio increases, the inflation coefficient will decrease linearly.

NXN is the governance token of the DST chain and the mortgage proof-of-work token of the gateway DPOS node. The DST chain adopts the DPOS mechanism. Block generation and verification in the DST network are completed by the POS gateway node. The system will automatically select the one with the largest mortgage amount. The top 51 most stable nodes online are used as POS nodes (hereinafter referred to as gateway nodes). The gateway node is dynamic. If the gateway node goes offline and cannot be connected, it will be automatically replaced by other qualified backup nodes. Gateway nodes are dynamic to ensure that the network never dies.

Chapter 3 DST chain supporting service components

3.1 DST communication system

The DST chain can build one-to-one encrypted video instant messaging within the platform based on the main chain. Its most prominent features are decentralization, high-strength encryption, anti-traffic tracking, support for edge computing and digital identity DID. By being content-oriented, Attract users to actively participate in the platform, such as social chat, video conferencing, live interaction, topic creation, etc.

In one-to-one video instant messaging, users can express their freedom of speech without being affected by the platform. Based on the point-to-point communication method, the decentralized network structure has the characteristics of non-tampering. When applied to social platforms, it can re-establish the credit system between people.

At the same time, the unique encryption technology of the DST chain ensures the security of one-to-one video instant messaging data transmission. Encryption and privacy protection technology are the core foundation of the DST chain.

The DST chain can encrypt user privacy information that needs to be kept confidential to ensure that the information is only disseminated or shared among specific users.

DST uses ECC elliptic 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 public keys to the other party. Afterwards, during the communication process, the other party's public key will be used to encrypt a piece of randomly generated data, and then the encrypted data will be sent to the other party. The party receiving the information will use its own private key to decrypt the data , so this method can provide higher efficiency. Security, because even if hackers steal the public key, they cannot decrypt the communication content, and the communication information of everyone on the entire network is unique.

3.2 DST wallet payment system

The DST wallet system is powerful and is mainly developed based on the technical infrastructure of mainstream wallets. It has also passed the security audit certification of CertiK. The wallet design pursues the ultimate user experience, making the payment environment safer, making asset management more convenient, and supporting multiple currencies. 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 tens of thousands of stress tests, and has powerful anti-theft technology to maximize the security of users' digital assets.

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

Security: The security of DST wallet is based on cryptography technology. Each transaction undergoes strict encryption verification to ensure the security of the transaction. In addition, due to the decentralized nature of blockchain, no one can

disrupt the entire payment system by attacking a single central authority.

Transparency: The transaction information of the DST chain is publicly recorded on the blockchain and can be viewed by anyone. This transparency promotes trust and reduces the risk of fraud.

Speed and efficiency: Blockchain payments can complete transactions faster and more efficiently than traditional bank transfers because blockchain does not require an intermediary to verify transactions.

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

3.3 Evm virtual machine module

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

Multi-language support: EVM can run smart contracts in multiple programming languages, including Solidity, Vyper, Lity, Bamboo, etc. This allows developers to use their best programming language to write smart contracts for the DST chain .

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

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

Higher interoperability: EVM 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, point-to-point

distributed application distribution system, with the goal of promoting 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 use different underlying blockchain development platforms and consensus mechanisms, or issue their own tokens (general tokens based on the same blockchain platform can also be used).

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

Applications developed based on Ethereum through side chains 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 using the organizational form of Dao (Decentralized Autonomous Organization), that is, the entire blockchain network can be updated through voting, advocating that everyone has the right to own and control their own digital identity, which can safely store themselves personal data and protect privacy. In a DST framework, individuals can have one digital identity that contains all this information and is managed individually, rather than having multiple digital identities managed by multiple centralized providers. Management is codified, programmed and automated. "Code is law" (code is law), organizations are no longer pyramidal but distributed, power is no longer centralized but decentralized, management is no longer bureaucracy but community autonomy, and organizational operations are no longer Companies are needed but replaced by highly autonomous

communities. In addition, because DAO (Decentralized Autonomous Organization) operates under operating standards and collaboration models 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 extensible 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 identity number, and the DST number can be generated without being connected to 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 DST networks. Through the DST node network, each number can easily interact with each other through text, voice, images, files, etc.

The job of a DST number is to map human-readable numbers (like "18888888888") to machine-readable identifiers (like DST addresses, other cryptocurrency addresses, content hashes, and metadata). FNS also supports "reverse resolution",

which can associate metadata such as canonical names or interface descriptions with DST addresses.

Top-level number segments, such as "1888888XXXX" and "1999999XXXX", are held by DPOS validators called communication gateways, which specify the rules that govern the allocation of their number segments. Anyone can obtain ownership of a DID for their own use according to the rules stipulated in these gateway contracts.

The DST number is the identity credential in the future decentralized society. It is owned and controlled by the individual. It also solves other needs for identity, such as the confirmation, verification, storage, management and use of personal data.

Chapter 4 Technical Support

DST main chain SDK

This SDK includes auth, bank, capability and other modules. 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 verification

The auth module is responsible for specifying the application's basic transaction and account types, as the SDK itself is agnostic to these details. It contains AnteHandler, which performs checking the validity of all basic transactions (signatures, nonces, auxiliary fields, etc.), and exposes the account keeper to allow other modules to read, write and modify the account.

(1) Concept

gas & fees

For network operators, fees serve two purposes:

Fees limit the growth of the state stored by each full node

and allow universal review of transactions with little economic value. Fees are best used as an anti-spam mechanism in situations where the validator has no interest in network usage or identity.

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

fees = ceil(gasLimit * gasPrices)

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

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

When adding a transaction or an idle transaction to the mempool, a validator checks whether the transaction's gas price (determined by the provided fees) matches any of the

validator's minimum gas prices. In other words, a transaction must offer at least one denomination of fees that matches the validator’s lowest gas price.

Tendermint does not currently provide fee-based mempool prioritization. Fee-based mempool filtering is node-local and is not part of the consensus. But with the minimum gas price set, such a mechanism can be implemented by node operators.

Because the market value of tokens will fluctuate, validators need to dynamically adjust their minimum gas prices. Since the market value of tokens will fluctuate, validators are expected to dynamically adjust their minimum gas prices to a level that encourages use of the network.

(2)Status

Account

Accounts contain authentication information for external users that is uniquely identifiable on the SDK blockchain, including public key, address, and account number/serial number for replay protection. For efficiency, the account structure also stores the user's balance as

`sdk.Coins' since the account balance must also be retrieved to pay fees.

Module clients that wish to add more account types can do so: Accounts are exposed to the outside world as an interface and stored internally as basic accounts or vested accounts.

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 accounts to storage, an

account keeper is required.

Account I is an interface for storing currency at a given address within the state. It assumes a concept of sequence numbers for replay protection, an account number concept for replay protection of 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

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

BaseAccount defines a basic account type. It contains all necessary fields for basic account functionality. Any custom account type should extend this type to implement additional functionality (for example, 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 does not have its own transaction handler, but exposes a special `AnteHandler' that performs basic validity checks on the transaction so that it can be thrown from the memory pool. Note that the ante handler is called on `CheckTx', but also on `DeliverTx', since Tendermint sponsors currently have the ability to include transactions in blocks they initiate that cannot pass `CheckTx'.

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 := account.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 only exposes 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 with the ability to read and write all fields for 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 iteraiton 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

}

Authorization(vesting)

Example

Simple example

Given a continuously vested account with 10 vested coins.

OV = 10

DF = 0

DV = 0

BC = 10

V = 10

V' = 0

Receive 1 coin immediately

BC = 11

Time has passed, 2 coins of vest

V = 8

V' = 2

Delegate 4 coins to validator A

DV = 4

BC = 7

Send 3 coins

BC = 4

More time passed and 2 more coins were delivered

V = 6

V' = 4

Sent 2 coins. At this point, the account cannot send anymore until more coins are vested or more coins are received. However, it can still be delegated.

BC = 2

Advanced examples

Same initial conditions as for the simple example.

Time passes, the ownership of 5 coins

V = 5

V' = 5

Delegate 5 coins to validator A

DV = 5

BC = 5

Delegate 5 coins to validator B

DF = 5

BC = 0

Validator A is slashed by 50%, making the value delegated

to A now 2.5 coins

Deauthorize from validator A (2.5 coins).

DF = 5 - 2.5 = 2.5

BC = 0 + 2.5 = 2.5

Revoke authorization from validator B (5 coins). At this point the account can only send 2.5 coins unless it receives more coins or until more coins are vested. However, it can still be delegated.

dv = 5 - 2.5 = 2.5

df = 2.5 - 2.5 = 0

bc = 2.5 + 5 = 7.5

Please note that we have an overdose of DV.

parameter

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

(1) Concept

Authorize

Any specific authorization type defined in the x/authz module must satisfy the authorization interface outlined below. Authorization determines exactly which permissions are granted. They are extensible and can be defined for any Msg service method, even outside the module in which the Msg method is defined. New ServiceMsg type authorized to use ADR 031.

Built-in authorization

The authz module comes with the following grant types

Send authorization

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

GenericAuthorization

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

Gas

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

(2)Status

Grants are identified by combining 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, message processing for the authz module is described.

Message/Approval Authorization

Use the MsgGrantAuthorization message to create an authorization grant.

This message is expected to fail under the following circumstances.

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

2. The provided expiration time is less than the current unix timestamp.

3. The authorization provided is not implemented.

Message/Revocation of Authorization

Allowed authorizations can be removed via the MsgRevokeAuthorization message.

This message is expected to fail under the following circumstances.

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

Message/Execution Authorization Request (MsgExecAuthorizedRequest)

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

This message is expected to fail under the following circumstances.

1. No authorization is implemented for the message provided.
2. The authorized person does not have the authority to run transactions.

3. The approved authorization has expired.

(4)Event

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 tracking special cases of spurious transfers that have to work differently with specific types of accounts (especially delegated/non-delegated vested accounts) . It exposes several interfaces with different capabilities for secure interaction with other modules that must change the user's balance.

Additionally, the Banking module tracks and provides query support for 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 many exported keeper interfaces, which can be passed to other modules that read or update account balances. Modules should use the least privileged interface to provide the functionality they require.

(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)Event

The bank module releases the following events

MsgSend, MsgMultiSend

In addition to handler events, the bank keeper will generate events when the following methods are called (or whatever method 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-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, considered the first owner of the capability.

Capabilities can be claimed by other modules as owners. `ClaimCapability' allows a module to claim a capability key it obtained from another module so that future calls to `GetCapability' will succeed. If a module that receives a

capability wishes to access it by name in the future, it must call `ClaimCapability'. Also, capabilities are multi-owner, so if multiple modules own a capability reference, 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 executing transaction, but will not be able to access it afterwards.

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

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

(2)Status

Index

CapabilityOwners

Capability

4.5 Crisis

The crisis module stops the blockchain if its invariants are broken. Invariants can be registered with the application during application initialization.

(1)Status

Fixed costs

Due to the high expected gas cost of validating a variable (and the potential to exceed 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 greater than the expected gas cost of running the constant with standard gas consumption methods. ConstantFee parameters are saved in the global params store.

(2)Message

MsgVerifyInvariant

The immutability 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

- Invariant routes are not registered

This message checks the provided immutability, and if the immutability is violated, it panics, stopping the operation of the blockchain. If immutability is broken, the fixed fee will never be deducted because the transaction is never submitted to the block (equivalent to being refunded). However, if the invariant is not broken, the fixed fee will not be refunded.

(3)Event

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 distribution 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. Collected rewards are pooled globally and passively distributed to validators and delegators. Each validator has the opportunity to charge delegators a fee for the rewards collected on behalf of the delegators. Fees are collected directly into the global reward pool and the validator proposer reward pool. Due to the nature of passive accounting, the

withdrawal of rewards must also occur whenever the parameters affecting the reward distribution rate change.

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

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

1. Atomic provisions for socially allocated multi-token fee proposer reward pool expansion

2. Commission fees for all rewards validators receive for their delegator stake are pooled together 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 Proof-of-Stake (PoS) blockchains, rewards earned from transaction fees are paid to validators. The fee distribution module distributes rewards fairly among the validator's constituent delegators.

Rewards are calculated on a periodic basis. This period is updated every time a validator's delegation changes, for example when a validator receives a new delegation. The reward for an individual validator can then be calculated by taking the total reward for the period before the delegation started, minus the current total reward.

Commissions are paid to validators when they are removed or when a validator requests a withdrawal. Commission is calculated and incremented on 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 globally tracked allocation parameters are stored in `FeePool`. Rewards are collected and added to a reward pool and distributed to validators/authorizers from here.

Note that the reward pool holds decimal coins (`DecCoins`) to allow fractional coins to be obtained from operations such as inflation. When coins are distributed from the reward 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 validator assignment information of the relevant validators will be updated every time.

The amount authorized to a validator is updated, the validator successfully proposes a block and is rewarded, any delegator withdraws 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 withdrawal fee. Because a delegation must withdraw fees (aka bonded tokens, etc.) when its properties change, its properties will remain the same, and the delegation's _accumulation_ coefficient can be calculated passively, only needing to know the last withdrawal height and its current properties.

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 each "BeginBlock", all fees received for the previous block are transferred to the assigned "ModuleAccount" account. When a delegator or validator withdraws their rewards, they do so from the `ModuleAccount'. During the start of the block, the different claims on the fees charged are updated below.

- Block proposers and their delegators of the previous

height receive a fee reward of 1% to 5%.

- Reserve community tax is charged.

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

To incentivize validators to wait for and include additional pre-commits in blocks, block proposer rewards are calculated from Tendermint pre-commit messages.

(4)Message

Set the withdrawal address (MsgSetWithdrawAddress)

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

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

The withdrawal address cannot be an account in any module. These accounts are added to the `blockedAddrs' array of allocation holders on initialization and are thus blocked 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 can be 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 beginning period.

- Validator period is increased. Because the validator's power and stake allocation may have changed, the validator's period is increased.

- The principal's reference count for the starting period is subtracted.

After

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

(6)Event

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 unrelated to certain types of evidence. An additional `ValidatorEvidence' extending `Evidence' has also been created to define an evidence contract 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. Evidence state is also stored and exported in GenesisState of the x/evidence module.

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 through 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 to

Allows for a flexible inflation rate, determined by market

demand, targeting a specific bond-to-equity ratio that balances market liquidity and stock supply. The moving rate of change mechanism ensures that if the staking percentage exceeds or falls below the target staking percentage, the inflation rate will adjust to further incentivize or disincentivize staking. Setting the target % bond rate below 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 staking percentage exceeds or falls below the target staking percentage, the inflation rate will adjust to further incentivize or disincentivize staking.

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

If the target adhesion rate is maintained at 67%, then the inflation rate will remain unchanged.

If the inflation rate is higher than the target bond rate, the inflation rate will decrease until it reaches a minimum value.

(2)Status

miner

Miner is a space used to save current inflation information.

Minter: 0x00 -> ProtocolBuffer(minter)

parameter

Mining parameters are saved in global parameters.

(3)Initial block

Minting parameters are recalculated at the beginning of each block and inflation is paid.

(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)Event

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 store.

There are two main types, Keeper and Subspace. A subspace is a separate namespace for a parameter store, where the keys are prefixed with a preconfigured space name. Keeper has one permission to access all existing spaces.

Subspace can be used by a single keeper, they require a private parameter storage space that cannot be modified by other keepers. The params Keeper can be used to add routes to the `x/gov` router to modify any parameters when the proposal is passed.

(1)Keeper

During the application initialization phase, you can use

Keeper.Subspace to allocate subspaces for the keepers of other modules and store them in Keeper.space. These modules can then obtain a reference to their specific parameter storage via Keeper.GetSubspace.

(2)Subspace

Subspace is a prefixed subspace of parameter storage. 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 the validators of the previous block, which is called the LastCommitInfo provided by Tendermint. As long as the LastCommitInfo contains a pre-commitment of +2/3 of the total voting power, it is valid.

Proposers are incentivized to include pre-commitments from all validators in the Tendermint LastCommitInfo by charging an additional fee proportional to the difference between the voting power included in the LastCommitInfo and

+2/3 (see fee allocation).

typeLastCommitInfo struct {

Round int32

Votes []VoteInfo

}

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

(2)Message

Unbind

If a validator has been automatically unbound due to outage and wishes to come back online and possibly rejoin the bound set, it must send MsgUnjail.

// MsgUnjail is an sdk.Msg, used to unbind the bound validator so that it returns to

// They return to the bound set of validators so they can start receiving provisions

// award.

message MsgUnjail {

string validator_addr = 1;

}

If a validator has enough stake to enter the top n = MaximumBondedValidators, it will automatically be rebonded and all delegators still delegated to the validator will be rebonded and start collecting provisions and rewards again.

(3)Initial block

Effectiveness tracking

At the beginning of each block, we update each validator's ValidatorSigningInfo and check if they have crossed the liveness threshold in a sliding window. This sliding window is defined by SignedBlocksWindow, and the index of this window is determined by the IndexOffset in the validator's ValidatorSigningInfo. For each block that is processed, regardless of whether the validator signs or not, the IndexOffset is incremented. Once the index is determined, 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 at which we can determine validity, minHeight.

If the current blocks are greater than minHeight, and the validator's MissedBlocksCounter is greater than maxMissed,

they will be deleted by SlashFractionDowntime, jailed by

DowntimeJailDuration, and the following values will be reset: MissedBlocksBitArray, MissedBlocksCounter, and IndexOffset.

(4)Hooks

cut hook

The reduction module implements StakingHooks defined in x/staking, which is used to record validator information. During the application's initialization process, these hooks should be registered in the staking module's structure.

The following hooks affect cut status.

AfterValidatorBonded creates a ValidatorSigningInfo

instance as described below.

AfterValidatorCreated stores a validator’s consensus key.

AfterValidatorRemoved removes a validator's consensus key.

Bound validator

After successfully binding a new validator for the first time, we create a new ValidatorSigningInfo structure for the now bound validator, which is 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)Event

The slashing module releases 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) Cut the 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 when re-joining the validator is not allowed set. However, due to consensus errors and the nature of ABCI, there can be a delay between the occurrence of a violation and the proof of the violation reaching the state machine (this is one of the main reasons for the existence of unbound periods).

(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. In this system, holders of the chain's native stake tokens can become validators and can delegate tokens to validators, ultimately determining the system's valid validator set.

(1)Status

State

astTotalPower tracks the total number of bonded tokens recorded during the previous end block. 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 lofting module.

Validator

A validator can have one of three statuses:

1. Unbound: The validator is not in the active set. They can't

sign blocks, and they can't earn rewards. They can receive delegations.

2. Binding": After validators receive enough binding tokens, they will automatically join the active set during EndBlock and their status will be updated to "Bound". They are signing and receiving rewards. They can get further Delegations. They may be slashed due to misbehavior. A validator that unbonds its delegator must wait for the duration of UnbondingTime (a chain-specific parameter), during which if the source validator's illegal behavior causes the token to be If it happens within a certain period, they can still cut it off.

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

Delegation

A delegate is identified by combining the DelegatorAddr (the delegate's address) with the ValidatorAddr. Delegators are indexed in storage as follows:

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

Delegator Shares

When a token is assigned to a validator, they are assigned a certain number of delegator shares based on a dynamic exchange rate, calculated based on the total number of tokens delegated to the validator and the number of shares issued to date:

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

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

(2)State Transitions

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

not bound to bound

When the validator's ranking in the ValidatorPowerIndex exceeds that of the LastValidator, the following transition will occur.

Set validator.Status to Bonded

Send validator token from NotBondedTokens to BondedPool ModuleAccount

Delete existing records from ValidatorByPowerIndex

Add new update record to ValidatorByPowerIndex

Updates the Validator object of this validator

Delete any ValidatorQueue records for this validator if present

Bound to unbound

Updates the Validator object of this validator

Set validator.Status to Unbonded

(3)Message

Msg/CreateValidator

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

initial delegate.

Msg/EditValidator

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

Msg/Delegate

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

Msg/Undelegate

Msg/Undelegate service messages allow principals to revoke their tokens from validators.

Msg/BeginRedelegate

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

(4)Initial block

Each time a sequential start block is called, historical

information is stored and pruned based on the HistoricalEntries parameter.

Historical information tracking

If the HistoricalEntries parameter is 0, BeginBlock performs no operation.

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 results in pruning one entry per block. However, if the parameter HistoricalEntries has been changed to a lower value, then there will be multiple entries in the storage that must be pruned.

(5) Final block

Each ABCI end block is called to update the queue and validator set operations specifying the changes to be performed.

Validator set changes

During this process, the staked validator set is updated through state transitions that run at the end of each block. As

part of this process, all updated validators are also returned to Tendermint for inclusion in the Tendermint validator set, which is responsible for validating Tendermint messages at the consensus layer.

queue

In equity, some state transitions are not instantaneous, but occur over a certain period of time (usually the bond-free time). When these transitions mature, certain 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 validator

When a validator is kicked out of the bound validator set (either through imprisonment or not having enough binding tokens), it starts the unbinding process, and all of its delegates also start unbinding (while still delegating to that validator). At this point, the validator is called an "unbound validator", by which point the validator will mature into an "unbound validator" after the unbinding period.

Each block of the validator queue will be checked for mature unbound validators (i.e. completion time <= current time, completion height <= current block height). At this point, any mature validators without any remaining delegates will be removed from the state . For all other mature unbound validators that still have remaining delegates, validator.Status switches from type.Unbonding to types.Unbonded.

Unconstrained delegation

Complete the unbinding of all complete entries in the UnbondingDelegations.UnbondingDelegations queue through the following process:

Transfer balance coins to representative's wallet address

Remove complete entries from UnbondingDelegation.Entries

If there are no entries remaining, remove the UnbondingDelegation object from storage.

Reauthorize

Unbinding of all mature reauthorizations is completed through the following process. Reauthorize an entry in the queue:

Remove mature entries from Redelegation.Entries

If there are no entries remaining, the reauthorization object is removed from the store.

(6)Hooks

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

①AfterValidatorCreated(Context, ValAddress)

Called when creating a validator

②BeforeValidatorModified(Context, ValAddress)

Called when the validator's status changes

③AfterValidatorRemoved(Context, ConsAddress, ValAddress)

Called when deleting a validator

④AfterValidatorBonded(Context, ConsAddress, ValAddress)

Called when binding validator

⑤AfterValidatorBeginUnbonding(Context, ConsAddress, ValAddress)

Called when the validator starts unbinding

⑥BeforeDelegationCreated(Context, AccAddress, ValAddress)

Called when creating a delegate

⑦BeforeDelegationSharesModified(Context, AccAddress, ValAddress)

Called when modifying the commission share

⑧BeforeDelegationRemoved(Context, AccAddress, ValAddress)

Called when deleting a delegate

(7)Event

This module opens 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"