DAO Secret Telegram

Decentralized secret telegram cluster based on blockchain technology

DST DAO · 2022

Foreword

People's communication in modern society is monopolized by various communication giants. Under many established rules, people gradually lose their freedom of communication; and traditional communication is expensive and easy to be eavesdropped and cracked, but vested interest groups are unwilling to upgrade technology to solve these problems. The problem is that this keeps the cost of personal communication high, and privacy is frequently leaked, and the security of people's communication cannot be guaranteed.

DST adheres to the belief of complete decentralization , is committed to promoting the innovation of blockchain communication technology, and at the same time upholds the concept of liberalism . For this reason, DST adopts a completely open source method and uses decentralized deployment technology. Anyone can apply to become a communication gateway on DST to provide other members with an entrance to the DST communication gateway .

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

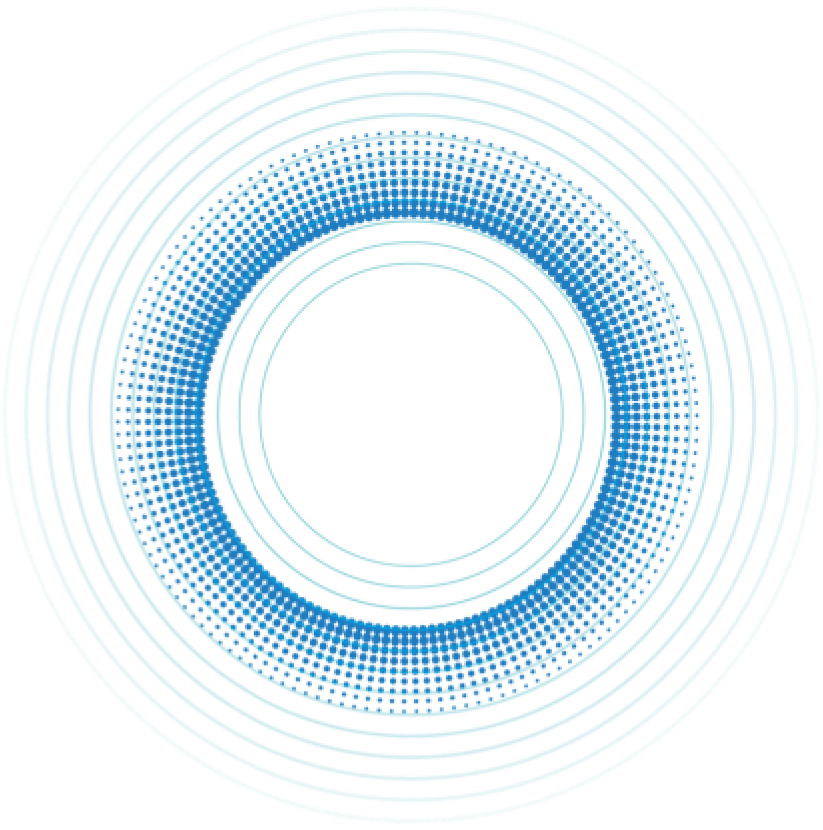
DST is the first distributed and decentralized free communication community with its own income-generating capabilities ;

DST has built a community network with real 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 to communicate, and use technologies such as asymmetric encryption of communication information to support and solve the problem. The privacy and security of personal communication , information leakage , communication trust and other issues have been solved .

At the same time, community members can also obtain the globally unique 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 disadvantages of high investment in the traditional communication industry . The cost of the entire communication industry has been greatly reduced, thus truly benefiting mankind and the communication industry through blockchain encrypted communication technology.

There is no central node and hierarchical management structure in the DST network . It achieves organizational goals through interaction, competition and collaboration between network nodes from bottom to top. Therefore, the business transactions 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, and are determined by each other's resource endowments, complementary advantages, and interests . Driven by a win-win situation. Each organization node will cooperate effectively under the incentive mechanism of the certificate according to its own resource advantages and talents, thus generating a strong synergy effect.

Relying on codes and smart contracts and the operating rules in DST, the responsibilities and rights of participants, and reward and punishment mechanisms are all open and transparent. In addition, through a series of efficient self-government principles, the rights and interests of relevant participants are precisely differentiated and balanced, that is, to match the corresponding rights and benefits to those individuals who work hard, make contributions, and assume responsibilities, so as to promote industrial division of labor and rights and responsibilities , Equal interests, making the organization more coordinated and orderly.



CONTENTS

Chapter 1 Blockchain Technology and Encrypted Social

1.1 Blockchain Technology and Web3.0

Blockchain is a decentralized distributed ledger technology, which 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 the basis of blockchain technology, they are not controlled by any central authority, and are fully determined jointly by their users and participants. This decentralized feature makes the Web 3.0 world more equal and just, removes the control of power by the central authority, and gives 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 a central authority to protect our data security, but this centralized approach is easy to be hacked and abused.

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 the web Internet

Web 1.0: The Read-Only Internet

Platform creation, platform ownership, platform control, and platform benefit

In the era of Web 1.0, in order to make it easier for people to share information, 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 changed from text to pictures to videos. But this information is static and read-only, and users can only be spectators and cannot participate in it.

Web 2.0: Read and Write the Internet

User creation, platform ownership, platform control, platform distribution

In the era of Web 2.0, its model will be more user-centric, allowing users to be creators of content, and interact and collaborate through social media. does not belong to the user himself.

Web 3.0: Internet of Value

User Created, User Owned, User Controlled, Protocol Assigned

In the era of Web 2.0, the development of the Internet has deviated from the original design track. As users change from information recipients to participants, users' digital behaviors on the network are also recorded one by one, and are held by various companies in the form of data. For example, Google records your search traces, Facebook and Twitter record your relationships and interactions, Amazon records your purchase records, and so on. These technology giants gain profits through a large amount of data, and algorithms control your world.

And the most important thing is that your personal data does not belong to you, but it has become a profit tool centered on various companies. For example, if Facebook goes bankrupt one day and closes your social account, then all the dynamics you posted in the past and all the relationships you established will disappear with the disappearance of Facebook, and your information does not belong to you in essence .

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 maintain ownership of the content they contribute, and can also get 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 hardened Web2.0, Web3.0's "readable + writable + ownable" Internet 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 the society, economy and politics of the future.

decentralized future

Blockchain and Web 3.0 will bring about a decentralized future that will allow everyone to participate in decision-making and governance. This will lead to a more equal and just 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 about a more secure and transparent network environment, which will protect users' data and privacy, and reduce hacking and abuse. At the same time, blockchain and Web 3.0 will also improve the transparency and fairness of the network, so that everyone can understand and join the Web3.0 network .

A more inclusive financial and economic environment

Blockchain and Web 3.0 will lead to 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, enabling everyone to participate in politics and governance. This will lead to a more democratic and just political environment with less abuse of power and less 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 Communication

Communication is the basis 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 the communication carrier of 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 have a subversive effect on the traditional Internet. As far as communication is concerned, its combination has natural advantages and common value appeals.

Blockchain technology can help communications achieve greater security. Security is paramount when communicating. 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 communication 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, allowing users to control their own communication data more autonomously. For example, point-to-point 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 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 network and cross-regional anonymous communication through the application of blockchain technology and the introduction of web3.0 protocol. Through the deep understanding and accumulation of the industry, as well as the insistence on decentralization belief and liberalism, the DST chain communication platform will lead an era of anonymous network with asset security and full freedom.

The biggest difference between the DST public chain and the traditional blockchain operation method is that DST implements a hybrid blockchain edge computing network built with mobile devices, PC devices, and server devices through a device interconnection routing architecture called NXN. The computing advantage of this network is that it can use equipment clusters to establish virtual network relays, and improve the stability of Dpos nodes through the dynamic committee algorithm mechanism, achieving 100 times the performance and throughput compared to mainstream public chains, and the consumption of handling fees is also lower. One ten-thousandth of the main chain, the reason is that the network built by mobile devices can meet the same network security strength as the mainstream public chain without additional investment, and it is also a public chain with communication transmission as the goal The indicators that the system must reach. Based on the Dpos pledge mode of the device cluster, the mobile terminal device only needs to perform the necessary data signature and communication relay network transmission, and does not need to synchronize the complete block and run a heavy calculation load, so it will not affect the normal use of the mobile terminal device. This is also the basis for the large-scale popularization of the DST network, and can make the DST network the world's largest provider of edge computing equipment and the largest decentralized communication service cluster.

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

DST users can use the built-in DID function to realize aggregate verification of Web3+Web2 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 revealing privacy.

As a global mainstream communication application, DST can realize the simultaneous interpretation function 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 start a barrier-free encrypted audio conversation 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 organization model, more like communism, the income generated will belong to the Dao organization collective, and the Dao organization manager will be responsible for allocating collective assets through voting, and the manager will safeguard the interests of all members and promote Dao organization development.

The following core features of DST

• DAO code governance : DST supports application voting upgrades, through which Dao organizations vote to decide whether to update, release and apply the public chain code, and determine the development framework of Dao application governance for DST.

• Enhanced interaction : DST achieves an excellent operating experience by transforming complex RPC block chain interaction instructions into a graphical mode. Compared with traditional block interaction, users hardly perceive that they are generating real-time data with the block chain. Interaction, for example, the user can click a bubble prompt in the chat window to complete the collection of Dpos rewards. This is an important advancement for the blockchain network operating system. It marks that traditional Internet users can enter the Web3 world through DST, and the command-line operations of pos pledge, pos share, pos contract, and pos revenue extraction, which were previously only operated by professionals, have progressed to a graphical interface.

• Decentralization: Traditional communication methods need to rely on centralized servers to forward information, while DST communication adopts a decentralized method for encrypted transmission, without third-party participation in the middle, which ensures the security and privacy of information, and at the same time avoids Service interruption due to server failure or attack .

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

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

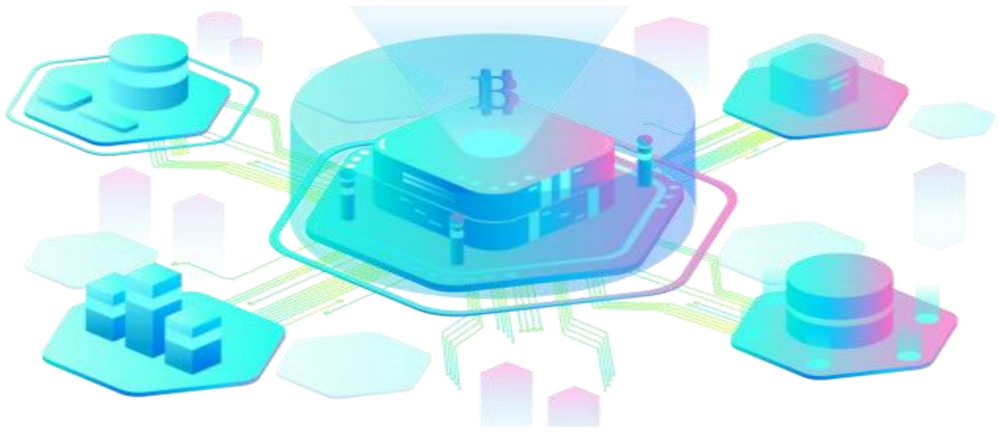
The DST blockchain communication platform strives to create a completely anonymous and untraceable encrypted communication protocol, and Token incentive model, to create a universal, complete support function, high performance, rich application scenarios, easy to use, and good user experience. , anonymous 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 general-purpose 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 provided based on edge computing nodes to achieve acceleration. , using the SDK, it can be integrated in any Android APP, and realize resource publishing, resource P2P loading capabilities, and finally realize data forwarding services that are far lower than the cost of IDC service providers.

Precautions:

- Support Android minimum SDK version: 23 or above for integration
- If you want to realize 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 Tokens

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. The rest of the tokens are obtained by POS pledge mining until the mining is exhausted.

2.2 Platform Tokens

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

2 . 3 Dpos market

In the blockchain network, if a device without proof of equity 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: equipment provider and token holder. Only through the combination of the two can a complete Dpos network be built, and both parties can also obtain mining rewards.

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

However, if Dpos nodes are frequently offline, it will reduce the stability of the blockchain network, so Dpos will establish a penalty mechanism to punish equipment holders who are unable to guarantee the operation of the equipment, and hardware equipment providers who want to hold DST tokens Users pledge their own nodes, and hardware device holders need to pledge certain tokens by 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, so that the Dpos node implements a distributed and highly available loose hardware architecture.

The users of the equipment cluster are displayed as a communication group. Through this form, communication, DAO governance, and POS market 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 according to the total amount of pledges and the minimum online equipment indicators, so that a larger and stable equipment cluster has a profit advantage.

Upgrade baseline for device clusters:

grade	Overall Minimum Pledge	least 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 DST Incentives and Outputs

Stake DST to the device cluster to get Dpos incentives, and the incentive ratio is obtained according to the inflation formula.

DST's daily output basis formula:

$$f(x) = \frac{Mining\ \ in\ \ the\ \ last\ \ 100\ \ days * DST\ \ mining\ \ inflation\ \ rate}{365}$$

The mining inflation rate for DST is:

$$DST\ mining\ \text{inflation rate} = \frac{35\% - Staking\ \ rate\ \ in\ \ the\ \ last\ \ 100\ \ days}{35\%} * \text{Median difference} + \text{Minimum inflation}$$

Pledge rate:

$$\text{Staking Rate} = \frac{Staking\ \ rate\ \ in\ \ the\ \ last\ \ 100\ \ days}{Mining\ \ in\ \ the\ \ last\ \ 100\ \ days}$$

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 the clusters, a cluster connectivity incentive algorithm was established, and the incentives were distributed to the cluster Dao governance pool in the form of DST. The elected committee Members make proposals and assignments.

Cluster connectivity incentive algorithm:

Staking amount defined at the cluster level * online rate * 5% * (communication traffic weight)

Cluster DAO governance incentive algorithm:

The new staking amount of the current cluster$*$ 10%

2 . 6 NXN incentives and output

In the early stage, NXN will only be issued in the form of airdrops. Airdrop targets include active test users, user nodes, and super nodes. By staking NXN, you can get mining incentives between 3% and 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. The block generation and verification in the DST network are all completed by the POS gateway node. The system will automatically select the most mortgaged, The top 51 nodes with the most online stability are used as POS nodes (hereinafter referred to as gateway nodes). The gateway node is dynamic. If the gateway node is disconnected 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 is based on the main chain and can build one-to-one encrypted video instant messaging within the platform. Its most prominent features are decentralization, high-intensity encryption, anti-traffic tracking, support for edge computing and digital identity DID, through content-oriented, Attract users to actively participate in the platform, such as community chat, video conferencing, live broadcast interaction, topic creation, etc.

In one-to-one video instant messaging, users can speak freely without being influenced by the platform. Based on the peer-to-peer communication method, the decentralized network structure has the characteristics of immutability. 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 realizes the security guarantee of one-to-one video instant messaging data transmission. Encryption and privacy protection technology is the core foundation of the DST chain. The DST chain can encrypt user privacy information that needs to be kept secret to ensure that information is only transmitted or shared among specific users.

DST adopts ECC elliptic curve encryption technology, which can encrypt your communication data with the best performance and strength. ECC belongs to the algorithm mode of asymmetric encryption and has been widely used in various data encryption scenarios.

Both parties use an asymmetric encryption algorithm to generate a pair of public key and private key , and send their own public key to the other party. Afterwards, during the communication process, a piece of randomly generated data will be encrypted with the public key of the other party, and then the encrypted data will be sent to the other party. The party receiving the information will decrypt the data with its own private key , so this method can provide higher security. 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

The DST wallet system is powerful. It is mainly developed based on the technical infrastructure of mainstream wallets. At the same time, it has passed the security audit certification of CertiK. The wallet design pursues the ultimate user experience, makes the payment environment safer, makes asset management more convenient, and supports multiple currencies. Multiple languages and multiple languages to create safe and easy-to-use digital asset management tools and entry-level applications. At the same time, it is easy to operate, tens of thousands of pressure tests, and powerful anti-theft technology to maximize the security of users' digital assets.

Decentralization: Blockchain technology makes the DST wallet payment system not require centralized institutions (such as banks), but 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 the DST wallet is based on cryptography technology, and each transaction is strictly encrypted and verified to ensure the security of the transaction.

Furthermore, due to the decentralized nature of the 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 anyone can view it. This transparency fosters trust and reduces the risk of fraud.

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

Cost saving: DST payment reduces the participation of intermediaries and reduces the cost of transactions. At the same time, because the transaction is decentralized, some additional fees and handling fees are also avoided.

3.3 Evm virtual machine module

The Evm module makes the DST chain highly scalable , 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 .

Support for multiple languages: EVM can run smart contracts in a variety of programming languages, including Solidity, Vyper, Lity, Bamboo, etc., which 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, authority management, etc., which 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 mechanism, etc., which enables 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, the 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, through which the download speed of DAPP can be 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 by themselves (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 it is also a basic service provider in the blockchain world. DAPP is to the blockchain, just like APP is to IOS and Android.

Through the side chain, the application developed based on Ethereum can pass DST The Dapp application market realizes 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 through voting, advocating that everyone has the right to own and control their own digital identity, which can be safely stored personal data and protect privacy. In the DST framework, instead of having multiple centralized providers manage multiple digital identities, individuals can have one digital identity that contains all of this information and manages it individually. Management is codified, programmed and automated. "Code is law" (code is law), the organization is no longer a pyramid but distributed, power is no longer centralized but decentralized, management is no longer bureaucratic but community autonomy, and organizational operations are no longer Corporations are needed and replaced by highly autonomous communities. In addition, because DAO (Decentralized Autonomous

Organization) operates under the operating standards and collaboration mode jointly determined by stakeholders, consensus and trust within the organization are easier to reach, 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, and 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 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 communicate with each other through text, voice, images, files, etc.

The job of a DST number is to map human-readable numbers (such as "18888888888") to machine-readable identifiers (such as DST addresses, other encrypted currency 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 blocks, such as "1888888XXXX" and "1999999XXXX", are held by DPOS validators called communication gateways, which specify the rules governing the allocation of their number blocks. Anyone can obtain the ownership of DID for their own use according to the rules stipulated by these gateway contracts.

The DST number is the identity credential in the future decentralized society, which is owned and controlled by individuals, and at the same time solves other needs of identity, such as confirmation, verification, storage, management and use of personal data.

Chapter 4 Technical Support

DST main chain SDK

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

4.1 Auth

Applied transaction and account verification

The auth module is responsible for specifying the application's basic transaction and account types, since the SDK itself is agnostic to these details. It contains AnteHandler, which performs checking of 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 accounts.

(1) concept

gas & fees

For network operators, fees serve two purposes

The fees limit the growth of the state stored by each full node and allow for generic censorship of transactions with little economic value. Fees are best suited as an anti-spam mechanism in situations where validators are not interested in network usage or identity.

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 transaction size. Operators should set a minimum gas price when launching their nodes. The gas unit cost for each token denomination you wish to support must be set:

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

When adding a transaction to the mempool or idling a transaction, the validator checks that the transaction's gas price (determined by the provided fees) matches any minimum gas price for the validator. In other words, a transaction must offer a fee of at least one denomination that matches the validator's minimum gas price.

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

Because the market value of tokens fluctuates, validators dynamically adjust their minimum gas prices. As the market value of tokens fluctuates, validators are expected to dynamically adjust their minimum gas price to a level that encourages usage of the network.

(2) status

account

Accounts contain authentication information for uniquely identifiable external users on the SDK blockchain, including public keys, addresses, and account numbers/serial numbers for replay protection. For efficiency, since the account balance must also be fetched to pay fees, the account structure also stores the user's balance as `sdk.Coins'.

Module clients who wish to add more account types can do so: accounts are exposed as an interface and stored internally as basic accounts or attributed 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 the account to storage, an account keeper is required.

AccountI is an interface for storing currency at a given address within a state. It assumes a notion of a sequence number for replay protection, a notion of an account number 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

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 basic account type. It contains all necessary fields for basic account functionality. Any custom account types should extend this type to implement additional functionality (eg, 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 does basic validity checks on a transaction so that it can be thrown from the mempool. Note that the ante handler is called on `CheckTx', but is also called on `DeliverTx', since Tendermint initiators currently have the ability to include transactions that fail `CheckTx' in their initiated blocks.

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 only exposes a 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 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 continuous vesting account with 10 vesting coins.

OV = 10

DF = 0

DV = 0

BC = 10

V = 10

V' = 0

Receive 1 coin instantly

BC = 11

time passed, vest for 2 coins

V = 8

V' = 2

Delegate 4 coins to validator A

DV = 4

BC = 7

give out 3 coins

BC = 4

More time passes and 2 more coins are delivered

V = 6

V' = 4

2 coins were sent. At this point, the account can no longer

send until more coins are vested or more coins are received. However, it can still be delegated.

BC = 2

Advanced example

The same initial conditions as for the simple example.

Time passes, 5 coins belong

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% such that the value delegated to A is now 2.5 coins

Deauthorize from validator A (2.5 coins).

DF = 5 - 2.5 = 2.5

BC = 0 + 2.5 = 2.5

Withdraw authorization (5 coins) from validator B. 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

Note that we have an excess 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 an SDK module that allows granting arbitrary permissions from one account (the grantor) to another account (the grantee). Authorization must be granted individually for a specific Msg service method using an implementation of the Authorization interface.

(1) concept

authorized

Any concrete 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.

Generic Authorization (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, 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 to each validator in both lists.

(2) status

Grants are identified by combining the grantor address (the byte address of the grantee), 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) news

The message processing for the authz module is described next

Message/Approve Authorization

Create a grant using the MsgGrantAuthorization message.

The message is expected to fail in the following cases.

1. Both the grantor and the grantee have the same address.
2. The expiration time provided is less than the current unix timestamp.
3. The authorization provided is not implemented.

Message / Withdraw Authorization

Allowed authorizations can be removed via the

MsgRevokeAuthorization message.

The message is expected to fail in the following cases.

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

Message/Execute Authorized Request (MsgExecAuthorizedRequest)

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

This message is expected to fail in the following cases.

1. No authorization was enforced for the message provided.
2. The authorized person does not have the authority to run the transaction.
3. 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 handles multi-asset coin transfers between accounts and tracks special cases of spurious transfers that must work differently with certain types of accounts (in particular delegated/non-delegated vested

accounts) . It exposes several interfaces with different capabilities for secure interaction with other modules that have to change user balances .

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 need.

(3) news

send message

```
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 handler events, the bank keeper will generate events when the following methods are called (or any method that eventually calls them)

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 persisted automatically; the calling module does not need to call `ClaimCapability`. Calling `NewCapability' will create the capability with the calling module and the name, as a tuple, considered as 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 got from another module so that future calls to `GetCapability' will succeed. If a module receiving a capability wishes to access it by name in the future, it must call `ClaimCapability'. Also,

capabilities are multi-owned, so if multiple modules own a capability reference, they will all own it. If a module receives a capability from another module without calling `ClaimCapability`, it can use it in the executed transaction, but it will not be able to access it afterwards.

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

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

(2) status

Index

Capability Owners

Capability

4.5 _ Crisis

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

(1) status

Fixed costs

Since the expected gas cost of validating a variable is very demanding (and has the potential to exceed the maximum allowed block gas limit), a constant fee is used instead of the standard gas consumption approach. The constant fee is intended to be greater than the expected gas cost of running the invariant with standard gas consumption methods. The ConstantFee parameter is stored in the global params store.

(2) message

MsgVerifyInvariant

The invariant 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 cover the flat fee
- Invariant routes are not registered

This message checks for the provided invariants, and if the invariants are broken, it panics, stopping the blockchain from functioning. If the invariant is broken, the fixed fee is never deducted because the transaction was never committed to the block (equivalent to being refunded). However, if the invariant 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 distribution mechanism describes a functional way to passively distribute rewards between validators and delegators . Note that this mechanism does not distribute funds as precisely as the active reward distribution mechanism, so it will be upgraded in the future .

The mechanism works as follows. Collected rewards are pooled globally and distributed passively among validators and delegators. Each validator has the opportunity to charge delegators a fee for rewards collected on behalf of delegators. Fees are collected directly into the global reward pool and the validator proposer reward pool. Due to the nature of passive accounting, whenever a parameter affecting the reward distribution rate changes, the withdrawal of the reward 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 bonded, unbonded, or re-delegated to an existing account, a full withdrawal of rewards must occur (because the rules of lazy accounting have changed). Whenever a validator chooses to change the rewarded commission, all accumulated commission rewards must be withdrawn at the same time.

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

1. Atomic provisions for socially distributed multi-token fee proposer reward pool inflation

2. The commission fee for all rewards earned by a validator for its delegator stake is pooled in a global pool, along with a validator-specific pool of proposer rewards. The mechanism used allows validators and delegators to withdraw their rewards independently and lazily.

(1) concept

In Proof-of-Stake (PoS) blockchains, rewards from transaction fees are paid to validators. The fee distribution module distributes rewards fairly among validators' constituent delegators.

Rewards are calculated on a periodic basis. This period is updated whenever a validator's delegation changes, for example, when a validator receives a new delegation . The reward for a single validator can then be calculated by subtracting the current total reward from the total reward for the epoch before delegation started .

Commissions are paid to validators when they are removed or when a validator requests a withdrawal. Commissions are calculated and incremented each BeginBlock operation to update the accumulated fee amount.

Rewards to delegators are distributed when delegators are changed or removed, or requested to withdraw. All slashes to validators that occurred during the current delegation are applied before the reward is distributed .

(2) status

Fee Pool

All globally tracked allocation parameters are stored in the `FeePool`. Rewards are collected and added to a reward pool , from where they are distributed to validators/authorizers.

Note that the reward pool holds decimal coins (`DecCoins`) to allow fractional coins to be earned 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

Validator assignment information for relevant validators is updated each time.

The amount authorized to the 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 value of 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 withdraw rewards
}
```

(3) Initial Block (Begin Block)

In each "BeginBlock", all fees received in the previous block are transferred to the assigned "ModuleAccount" account. When delegators or validators withdraw their rewards, they withdraw from `ModuleAccount'. The different claims to the fees charged during the start of the block are updated below.

- The block proposer and its delegators of the previous height receive a fee reward of 1% to 5%.

- Reserve community tax is charged.

- The remaining part is distributed to all bonded validators in proportion to voting rights

To incentivize validators to wait and include additional precommits in blocks, block proposer rewards are calculated from Tendermint precommit messages.

(4) news

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.

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

The withdrawal address cannot be the account of any module. These accounts are blocked from being withdrawal addresses by being added to the allocation holder's `blockedAddrs' array upon initialization.

```
func (k Keeper) SetWithdrawAddr(ctx sdk.Context, delegateAddr 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, delegateAddr, 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

- Delegate rewards are withdrawn to the delegator's withdrawal address. Rewards include the current period, not the started period.

- Validator period is increased. The validator's epoch is increased because the validator's power and stake allocation may have changed.

- The reference count of the principal's start period is subtracted.

After

The client's starting height is set to the previous epoch.

Due to the existence of the before-hook, this period is the last period for delegators to receive rewards.

(6) Events

Begin Blocker

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]

| bonus proposer reward|string(dec)|"0.040000000000000000"[0]

|withdrawaddenabled|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 to certain types of evidence. An additional `ValidatorEvidence', which extends `Evidence', is also created to define a proof contract against malicious validators.

```
                    Governancetype Evidence interface {
proto.Message
    Route() string
        Type() string
String() string
Hash() tmbytes. HexBytes
ValidateBasic() error
    // Height at which the injury 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 evidence of valid commits in the state. Evidence state is also stored and exported in GenesisState in 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 with equity supply. The moving rate-of-change mechanism ensures that if the staked percentage exceeds or falls below the target staked percentage, the inflation rate will adjust to further incentivize or discourage staking. Setting the target % bond rate below 100% encourages the network to keep some tokens unbonded which should help provide some liquidity.

The moving rate-of-change mechanism ensures that if the staked percentage exceeds or falls below the target staked percentage, the inflation rate will adjust to further incentivize or discourage 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 binding rate of 67% is maintained, then the inflation rate will remain unchanged.

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

(2) status

miner

A miner is a space used to hold 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) Events

Begin blocker

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 an independent namespace for a parameter store where keys are prefixed with a pre-configured space name. Keeper has a permission to access all existing spaces.

Subspaces can be used by a single keeper, and they require a private parameter storage space that other keepers cannot

modify. The params Keeper can be used to add routes to the `x/gov` router to modify any parameters when the proposal passes .

(1)Keeper

In the application initialization phase, you can use Keeper.Subspace to allocate subspaces for the keeper 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 for 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).

```
type LastCommitInfo struct {
Round    int32
Votes    [] VoteInfo
}
```

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

(2) message

unbind

If a validator was automatically unjailed due to downtime, 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 go back to the set of bound validators so they can start receiving rules
// award.
message MsgUnjail {
string validator_addr = 1;
}
```

If the validator has enough stake, before entering n = MaximumBondedValidators, it will automatically be rebonded , and all delegators still delegated to the validator will be rebonded and start collecting regulations and rewards again.

(3) Initial block

Validity Tracking

At the beginning of each block, we update each validator's ValidatorSigningInfo and check whether they have crossed the liveness threshold within 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 processed, IndexOffset is incremented regardless of whether the validator signed it or not. Once the index is determined, MissedBlocksBitArray and MissedBlocksCounter will be updated accordingly.

Finally, to determine if a validator has crossed the validity threshold, we take 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 block is greater than minHeight, and the validator's MissedBlocksCounter is greater than maxMissed, they will be deleted by SlashFractionDowntime, will be jailed for DowntimeJailDuration, and reset the following values: MissedBlocksBitArray, MissedBlocksCounter, and IndexOffset.

(4)Hooks

cut hook

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

The following hooks affect the slash state.

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) 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) Cut 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 time period that is not allowed to rejoin the validator set. However, due to consensus errors and the nature of ABCI, there may be a delay between when a violation occurs and when proof of the violation reaches the state machine (this is one of the main reasons why there is an unbound period).

(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 (equity)

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 effective set of validators for the system.

(1) Status

State

astTotalPower tracks the total amount of bound tokens recorded during the last 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 stakeout module.

Validator

A validator can have one of three states

1. Unbound: The validator is not in the active set. They can't sign up for 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 "Binding". They are signing and getting rewards. They can get further Delegation. They may be slashed due to misbehavior. Validators who unbond their delegators must wait for the duration of UnbondingTime (chain-specific parameter) happens within a certain period, they can still chop it off.

3. Unbinding: When a validator leaves the active set by choice, or due to slashing, sawing, or tombstoning, all their delegations will begin to unbind. All delegations then have to wait for the UnbondingTime before moving their tokens from the BondedPool to their accounts.

Delegation

A delegate is identified by combining a DelegatorAddr (the address of the delegator) with a ValidatorAddr, and the delegator is indexed in the store 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 assigned a certain number of delegator shares according to a dynamic exchange rate, calculated from the total number of tokens delegated to validators 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 delegators then undelegate, the amount of tokens they receive is calculated based on their current stake and back e.

(2) State Transitions (state transition)

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

unbound to bound

When the validator's rank in ValidatorPowerIndex exceeds LastValidator, the following transitions will occur.

Set validator.Status to Bonded

Send validator token from NotBondedTokens to BondedPool ModuleAccount

Delete existing records from ValidatorByPowerIndex

Add new updated record to ValidatorByPowerIndex

Update the Validator object for this validator

Deletes any ValidatorQueue records for this validator, if present bound to unbound

Update the Validator object for this validator

Set validator.Status to Unbonded

(3) news

Msg/CreateValidator

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

Msg/EditValidator

Commission rate can be updated using Msg/EditCandidacy service message.

Msg/Delegate

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

Msg/Undelegate

The Msg/Undelegate service message allows the delegator to revoke its token from the validator.

Msg/BeginRedelegate

The re-authorization command allows delegators to switch validators immediately. Once the binding period is lifted, reauthorization will be done automatically in EndBlocker.

(4) Initial block

Each time the sequential start block is called, the 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, while 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 store that must be pruned.

(5) Final block

Each abci end block call, an operation that updates the queue and validator set specifies the changes to perform.

Validator set changes

During this process, the set of staked validators 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 staking, certain state transitions are not instantaneous, but take place over a certain period of time (usually bond-free time). When these transitions mature, certain actions must be performed to complete the state operations. 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 bonded validator set (either by jailing or not having enough bonded tokens), it starts the unbonding process, and all its delegates also start unbonding (while still delegating to the validator). At this point, the validator is said to be an "unbound validator", by which the validator will mature after the unbond period to become an "unbound validator".

Each block in the validator queue will be checked for mature unbound validators (ie, completion time <= current time , completion height <= current block height). At this point, any mature validators that do not have any delegation remaining are removed from the state . validator.Status switches from type.Unbonding to types.Unbonded for all other mature unbonded validators that still have a remaining delegation .

unconstrained delegation

all complete entries in the UnbondingDelegations.UnbondingDelegations queue is done by the following process :

Transfer balance coins to the delegate's wallet address

Remove complete entries from UnbondingDelegation.Entries

Removes the UnbondingDelegation object from storage if no entries remain.

Reauthorize

Unbind all mature reauthorizations through the following procedure. Entries in the reauthorization queue:

Remove mature entries from Redelegation.Entries

If no entries remain, the reauthorization object is removed from storage.

(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 stubs:

①AfterValidatorCreated (Context, ValAddress)

Called when a validator is created

②BeforeValidatorModified (Context, ValAddress)

Called when the validator's state changes

③AfterValidatorRemoved (Context, ConsAddress, ValAddress)

Called when a validator is removed

④ AfterValidatorBonded (Context, ConsAddress, ValAddress)

Called when a validator is bound

⑤ AfterValidatorBeginUnbonding (Context, ConsAddress, ValAddress)

Called when the validator starts unbinding

⑥BeforeDelegationCreated (Context, AccAddress, ValAddress)

Called when a delegate is created

⑦BeforeDelegationSharesModified (Context, AccAddress, ValAddress)

Called when the delegate share is modified

⑧BeforeDelegationRemoved (Context, AccAddress, ValAddress)

Called when a delegate is removed

(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"